AD-A054 219 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 15/7

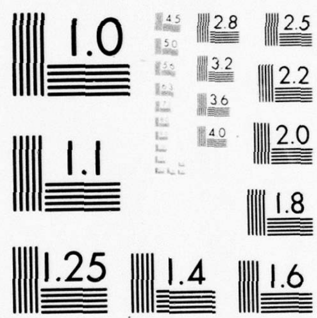
THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK). PROG--ETC(U)

APR 78

UNCLASSIFIED CCTC-SM-MM-9-74-V4-1-CH-3 NL

1 OF 3

AD
A054219



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

AD A054219

DEFENSE COMMUNICATIONS AGENCY
COMMAND AND CONTROL
TECHNICAL CENTER
WASHINGTON, D. C. 20301

FOR FURTHER TRAN

IN REPLY REFER TO: C314

APR 1978

TO: RECIPIENTS

SUBJECT: Change 3 to Program Maintenance Manual CSM MM 9-74, Volume IV, Sortie Generation Subsystem

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations.

2. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.

3. When this change has been posted, make an entry in the Record of Changes.

FOR THE DIRECTOR

286 Enclosures
Change 3 pages

J. DOUGLAS POTTER
Assistant to the Director
for Administration

AD No.
DDC FILE COPY

The CCTC Quick-Reacting General War Gaming System (QUICK). Program Maintenance Manual. Volume IV. Sortie Generation Subsystem. Change 3.

276 p.

CCTC-SM-MM-9-74-V4-1-CH-3,
CCTC-SM-MM-9-74-V4-2-CH-3

409 658

EFFECTIVE PAGES - JANUARY 1978

This list is used to verify the accuracy of CSM MM 9-74 Volume IV after change 3 pages have been inserted. Original pages are indicated by the letter O, change 1 pages by the numeral 1, change 2 pages by the numeral 2, and change 3 pages by the numeral 3.

Page No.	Change No.	Page No.	Change No.
Front Cover, Part I	0	303-313	0
Title Page	0	314-317	3
ii	1	318-328	0
iii-v	3	329-331	3
vi	1	332	0
vii-ix	3	733-736	1
x	0	Front Cover, Part II	1
xi	3	Title Page	0
xii	0	ii	1
1-5	3	iii	0
6	0	iv-v	3
7-8	3	vi	1
9	0	vii-xi	3
10	2	xii	0
11-29	3	333-337	3
29.1-29.2	3	338	0
30-35	3	339	2
36	0	340	0
37-160	3	341	2
161-162	2	342	0
163-164	0	343-350	3
165	3	351	2
166-170	0	352	0
171	3	353	2
172-178	0	354-355	0
179	3	356-357	2
180	0	357.1-357.2	2
181	3	358	2
182-184	0	359	3
185-186	2	360	2
187-188	0	361-370	3
189	2	371	0
190	0	372	2
191	3	373-375	0
192-272	0	376	3
273	3	377-379	0
274-296	0	380-390	3
297-299	3	391-392	0
300-301	0	393	3
302	3	394	2

Page No.	Change No.
394.1-394.2	2
395-397	2
398	3
399	2
400	3
401-405	0
406	3
407-423	0
424-427	3
428	0
429-430	3
431	0
432-437	3
438-439	0
440	3
441-442	0
443-450	3
451-466	0
467-477	3
478-480	0
481-482	3
483	0
484-485	3
486-497	0
498	3
499-501	0
502-505	3
506-523	0
524-525	3
526	0
527-528	3
529	0
530	3
531-542	0
543-545	3
546-547	0
548	3
549-553	0
554-558	3
559	0
560-563	3
564-567	0
568-569	3
570	2
571	0
572-578	3
579	0
580	3
581-587	0
588	3
589	0
590	3
591-610	0
611-614	3
615	1
616-619	2
619.1-619.2	2
620	2
621	0
622	2
623-627	1
628-644	0
645-661	3
662-666	0
667	3
668	0
669	3
670-732	0
733-736	1

CONTENTS

Part I

Part II

ILLUSTRATIONS (PART I)

TABLES (PART I)

ABSTRACT

The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, provide output summaries, and produce input tapes to simulator subsystems external to QUICK. QUICK has been programmed in FORTRAN for use on the CCTC HIS 6000 computer system.

The QUICK Program Maintenance Manual consists of four volumes: Volume I, Data Management Subsystem; Volume II, Weapon/Target Identification Subsystem; Volume III, Weapon Allocation Subsystem; Volume IV, Sortie Generation Subsystem. The Program Maintenance Manual complements the other QUICK Computer System Manuals to facilitate maintenance of the war gaming system. This volume, Volume IV is in two parts providing the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the programs and subroutines of the Sortie Generation subsystem. The associated program listings which are dynamic and voluminous are not contained herein. However, the program listings may be obtained by arrangement with the CCTC QUICK Project Officer. Companion documents are:

a. USERS MANUAL
Computer System Manual CSM UM 9-77, Volume I
Computer System Manual CSM UM 9-77, Volume II
Computer System Manual CSM UM 9-74, Volume III
Computer System Manual CSM UM 9-74, Volume IV
Provides detailed instructions for applications of the system.

b. TECHNICAL MEMORANDUM
Technical Memorandum TM 153-77
Provides a nontechnical description of the system for senior management personnel.

SECTION 1. GENERAL

1.1 Purpose

This volume of the QUICK Program Maintenance Manual describes the programs which are part of the QUICK Sortie Generation Subsystem. The information contained herein is presented on a program-by-program basis, complemented with discussions on computer programming maintenance on a subject-by-subject basis. The program-by-program discussions are structured so that a maintenance programmer can understand the program functions and programming techniques. The computer subjects are structured to inform the maintenance programmer of overall system programming techniques and conventions.

1.2 General Description

The Sortie Generation subsystem operates using the output from the Weapon Allocation subsystem, and produces detailed bomber and missile (delivery vehicle and weapon) sortie specifications. Thus, it accepts a near-optimal weapon allocation, and from this as well as consideration of delivery vehicle characteristics and other factors, generates a detailed plan of attack for one opposing side in a hypothetical general war.

The subsystem consists of programs FOOTPRNT, POSTALOC, PLANOUT, and PLOTIT, as shown in figure 1. Figure 2 shows the relationship of the Sortie Generation subsystem to other QUICK subsystems in terms of procedural and information flow.

In addition to the plan generation requirements, per se, the output of this subsystem is utilized alternatively by:

a. Damage Assessment systems external to QUICK which utilize weapon/target strike data (DGZ tapes) as required

b. General War simulation models external to QUICK (e.g., NEMO and ESP) which utilize relevant strike data as required (DGZ, A/B tapes and TABLTAPE).

As shown in figure 3, the Sortie Generation subsystem is initiated when a spill tape from program ALOCOUT (Weapon Allocation subsystem) is received as input for processing by program FOOTPRNT.* Program FOOTPRNT

*FOOTPRNT is used if the plan includes MIRV system. The ALOCGRP file is then used in POSTALOC in place of the TMPALOC file.

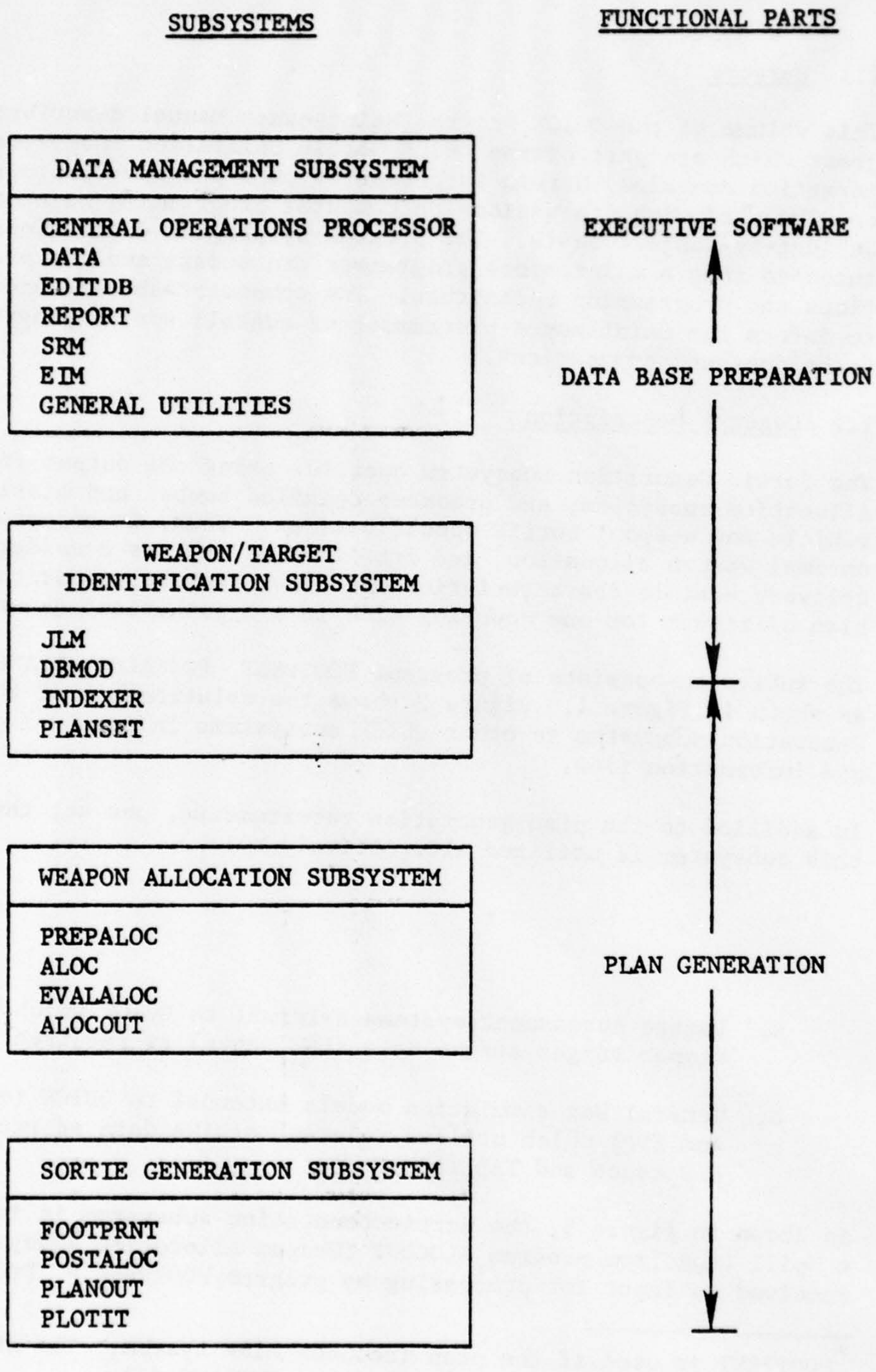


Figure 1. Major Subsystems of the QUICK System

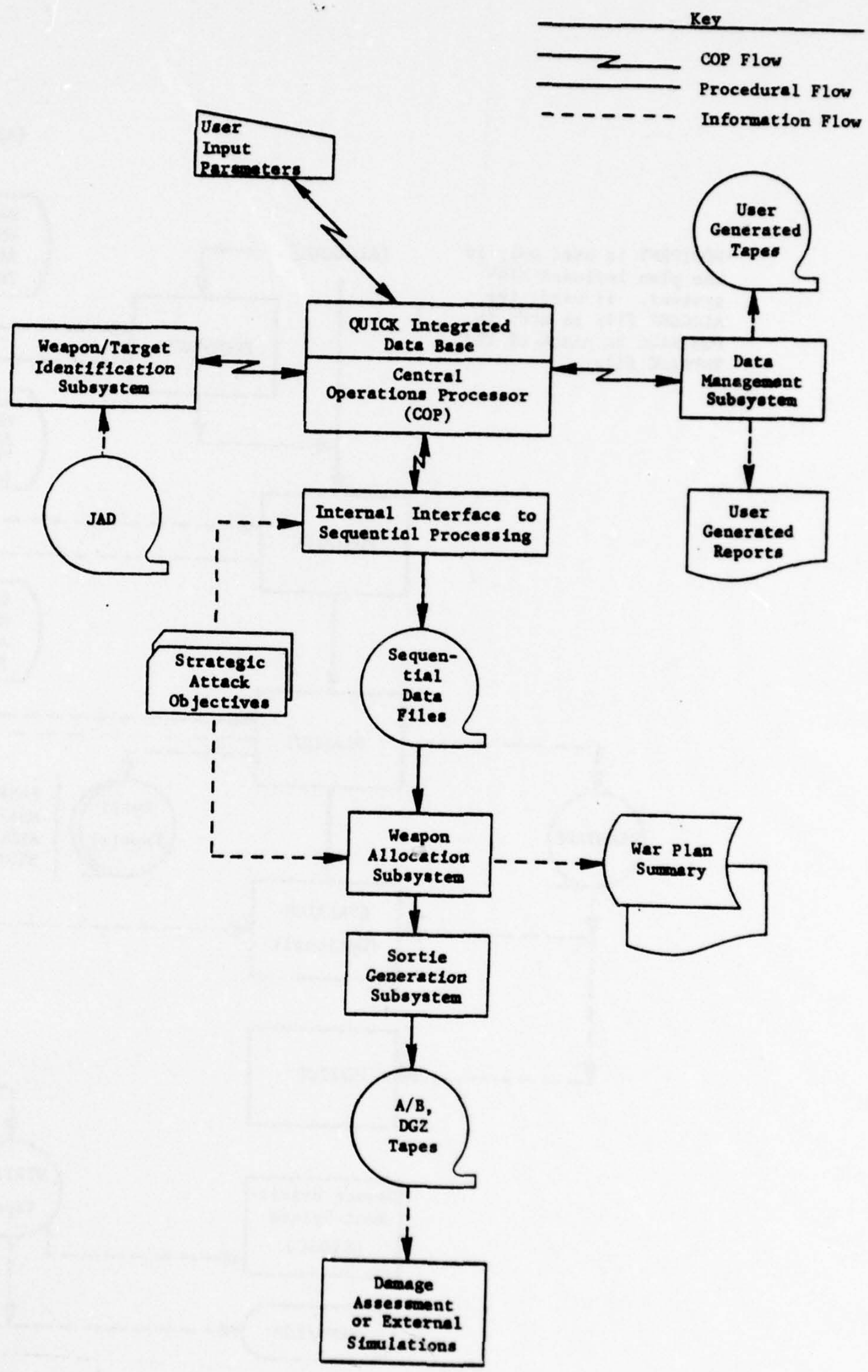


Figure 2. Procedure and Information Flow in QUICK/HIS 6000

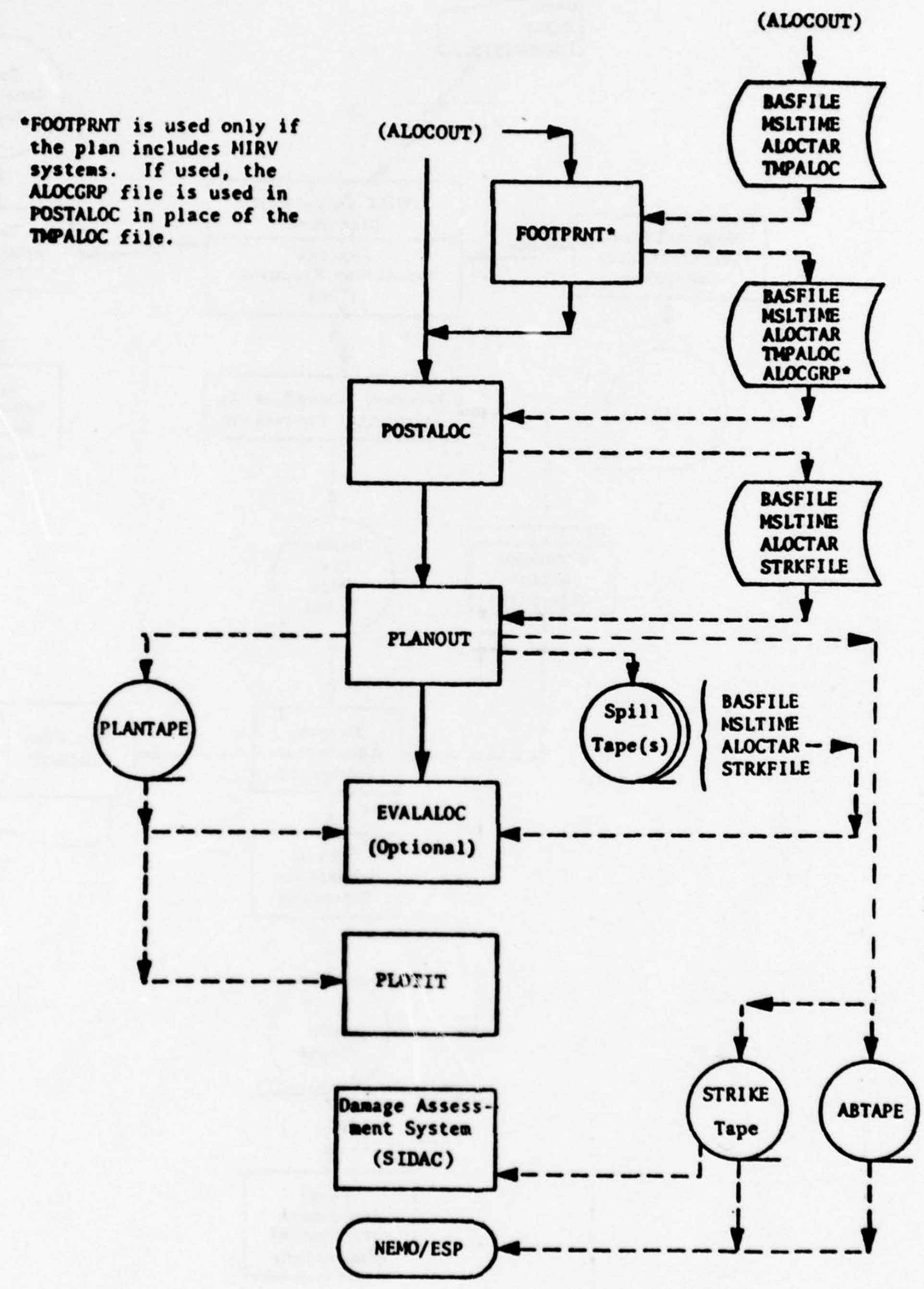


Figure 3. Sortie Generation Subsystem

reads the strike data from TMPALOC file. Information for bomber groups and for missile groups without a MIRV capability is copied directly to the ALOCGRP file, while the data for the missiles with MIRV payloads are processed to create detailed reentry vehicle-target point assignments which satisfy the various constraints. These detailed assignments are then also written onto the ALOCGRP file.

POSTALOC then reads ALOCTAR one weapon group at a time to develop specific specific bomber and missile sorties. The missile launch events prepared by POSTALOC are output to the STRKFILE. The bomber sorties are prepared one corridor at a time for each group and output to the STRKFILE.

The sortie plans at this point are not fully detailed. Program PLANOUT therefore adds the required data, e.g., timing information, and creates two output tape files. The plan tape PLANTAPE contains detailed mission plans in a form suitable for review or use in program PLOTIT and EVALALOC. In addition, A/B and strike tapes are prepared for use in subsystems external to QUICK.

Program PLOTIT uses the PLANTAPE tape generated from a QUICK data base to produce rectilinear plots of bomber or tanker sorties. Plots are produced for PIC-1 and PIC-2 maps, as well as, on the scales: 50 x 40 inches, 20 x 20 inches, or 10 x 10 inches. For plots other than PIC-1 and PIC-2 maps, the program will automatically choose the most desirable origin for the plots and will draw axes and table them accordingly. Up to 10 sorties may be plotted on one graph (or 200 events). The option is available to plot all sorties on the PLANTAPE or to plot only sorties specified on input cards.

SECTION 2. PROGRAM FOOTPRNT

2.1 <u>Purpose</u>

The purpose of program FOOTPRNT is to divide the set of targets assigned to a MIRV group into subsets, each of which is assigned to one booster in the group. This division is constrained by the limitations of the MIRV systems so that the acceptable booster assignments lie within a geographic pattern known as a footprint. The program is divided into two sections, the selection section and the assignment section.

The selection section interrogates the entire input target list and forms a definable subset of the targets that potentially may form a footprint. The selection process logically collects targets that are geographically located within a maximum footprint contour.

Given a collection of targets from the selection section, the assignment section applies dynamic programming techniques to the selected targets and attempts to form a footprint within the fuel constraints of the MIRV system.

This program receives the TMPALOC file from program ALOCOUT and prepares the ALOCGRP file. This latter file contains the weapon-target allocation ordered by weapon groups. Program FOOTPRNT processes only those groups with a MIRV capability. The target assignments to those groups are divided into subassignments, each of which is a feasible MIRV booster assignment.

The user input footprint assignment parameters define the nature of feasible footprints. These parameters define the fuel used in delivering a series of reentry vehicles and decoys in a specific geographic pattern. The required user-input parameters are a set of coefficients to equations used to model the physical MIRV systems.

2.2 Input Files

There are three basic sources of input to program FOOTPRNT: the TMPALOC file, the BASFILE file, and the user-input parameter cards. The TMPALOC file is produced by program ALOCOUT. This file contains the assignment of weapons to targets, ordered by group and corridor. That is, all targets assigned to weapons from the same group are placed together on the TMPALOC file. Within each group, the targets assigned to the same corridor are also placed together. Common /STRKSUM/ on this file describes the ordering of corridors and strikes for each group. For missile groups, only corridor 0 is used, so this latter ordering by corridor has no effect. Program FOOTPRNT processes only those missile groups with a MIRV capability. Therefore, all information on the TMPALOC file which does not deal with MIRV weapons is merely copied verbatim to the ALOCGRP file. For MIRV weapons, FOOTPRNT reads from the TMPALOC file the information contained in common blocks /STRKSUM/ and /C3/ as output by ALOCOUT. These data are transferred to common blocks /STRKSUM/ and /RAIDATA/ in program FOOTPRNT. This information defines the possible targets which can be assigned to the weapons in each MIRV group. Program FOOTPRNT performs the assignment of targets to individual boosters and outputs the information on ALOCGRP file in a form such that program POSTALOC can prepare the basic sorties for each MIRV missile booster.

The data retrieved from the BASFILE include: plan size information (/MASTER/), file information (/FILES/), weapon group information (from blocks /PAYLOAD/, /WPNTYPE/, /WPNGRP/ on BASFILE), and excess weapon assignment information (from /EXCESS/ on BASFILE).

The user-input parameters define the footprint constraint definition variables.

2.3 Output Files

Program FOOTPRNT produces the ALOCGRP file. (This file is written using the QUICK filehandler. The description of those routines describes the format of the physical makeup of this file.) The file is output to the disk unit for use by program POSTALOC. If there are no MIRV weapons in the plan, program FOOTPRNT copies the TMPALOC file onto the ALOCGRP file. If there are MIRV weapons in the plan, the information for those groups without MIRV weapons is copied verbatim onto the ALOCGRP file.

The only other data file required by program FOOTPRNT is the BASFILE. The program reads from this file:

a. Plan size information (/MASTER/)

b. Logical file units (/FILES/)

c. Weapon group information (/WPNGRPX/ in FOOTPRNT; /PAYLOAD/, /WPNTYPE/, /WPNGRP/ on BASFILE)

The program operates on a group-by-group basis. Each group is considered independently of all other groups. Hence, the discussion of all subroutines except the main program will consider only the operations required for the current group. For non-MIRV weapon groups, the TMPALOC data are copied onto ALOCGRP. For MIRV groups, further processing is required.

The program consists of two sections, the selection section and the assignment section. The selection section collects targets that potentially may form a footprint. The assignment section determines if from the list of potential targets, a footprint can be formed. If so, necessary bookkeeping reflects that fact and a new target collection processed. Figure 4 displays a functional diagram of FOOTPRNT.

The card input data for each module are discussed in detail in the subroutine section (subroutine TABLINPT). These data contain information on fuel loads, maximum ranges, and fuel consumption rates.

In essence, the operation of this program is a reordering of a list. The input is an unordered list of strikes assigned to the group. The required processing is to subset and reorder this list so that each sublist is a feasible booster assignment. Since much of the processing involves lists of various kinds, it is useful here to describe some of the basic lists that are involved in processing (input, RAIDATA).

The first list is the input data, contained in common /RAIDATA/, and common /GRPDATA/. The data consist of several lists, each containing one element for each target assigned to the group. The lists contain index number (INDEXNO), target latitude (TGTLAT), target longitude (TGTLONG), relative damage value (RVAL), offset latitude (DLAT), offset longitude (DLONG), fixed assignment indicator (LXLLFIX), target designator code (DESIG), target

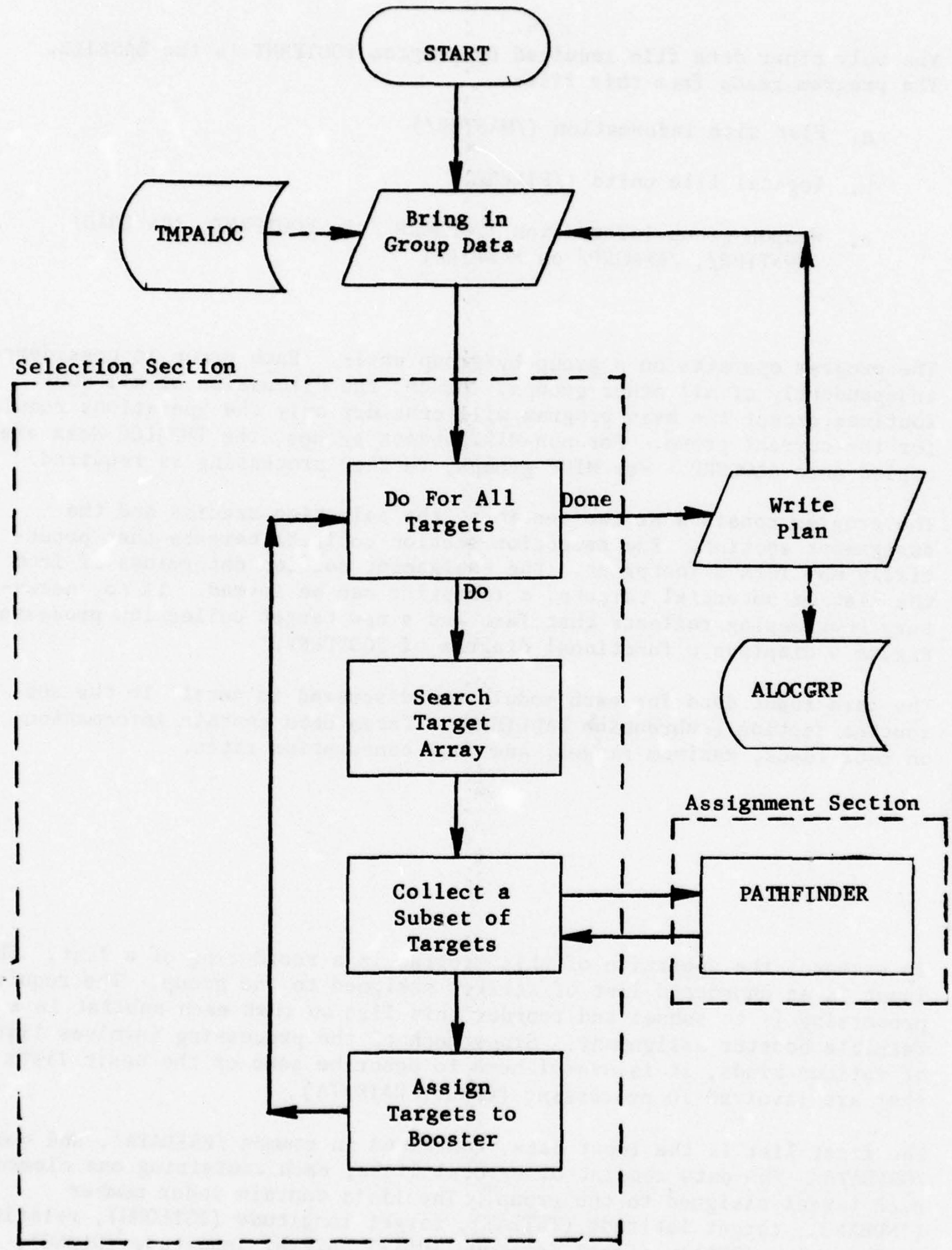


Figure 4 Block Diagram of Program FOOTPRNT

task/subtask code (TASK), target country location code (CNTRYLOC), target flag code (FLAG), salvo number (ISAL), and height of burst (LXIHOB). Most of these data are not needed after the early calculations, so the data are written out onto a scratch file.

The geographic data are then converted to polar coordinates centered on the group centroid with axis passing through the North Pole. The range and launch azimuth from centroid to each target is computed and stored. The data are then reordered according to increasing value of launch azimuth. (The sequence array required for this ordering is also written on the scratch tape.) The reordered targets are then processed.

CONTENTS OF THIS PAGE INTENTIONALLY DELETED

The details of the processing of elements is contained in subsequent parts of this section. Figure 5 shows the hierarchy and function of the major subroutines of this program.

2.5 Common Block Definition

This program references external common blocks /MASTER/ and /TAPES/ from the BASFILE. In addition, certain information for common block /WPNGRPX/ is read from the BASFILE blocks /PAYLOAD/, /WPNGRP/, and /WPNTYPE/.

Table 3 defines the variables in each common block.

Common blocks ITP, TWORD, MYIDENT, NOPRINT, IFTPRNT, FILABEL, and MYLABEL are FILEHANDLER common blocks and fully described in Program Maintenance Manual, Volume I, Data Assembly Subsystem.

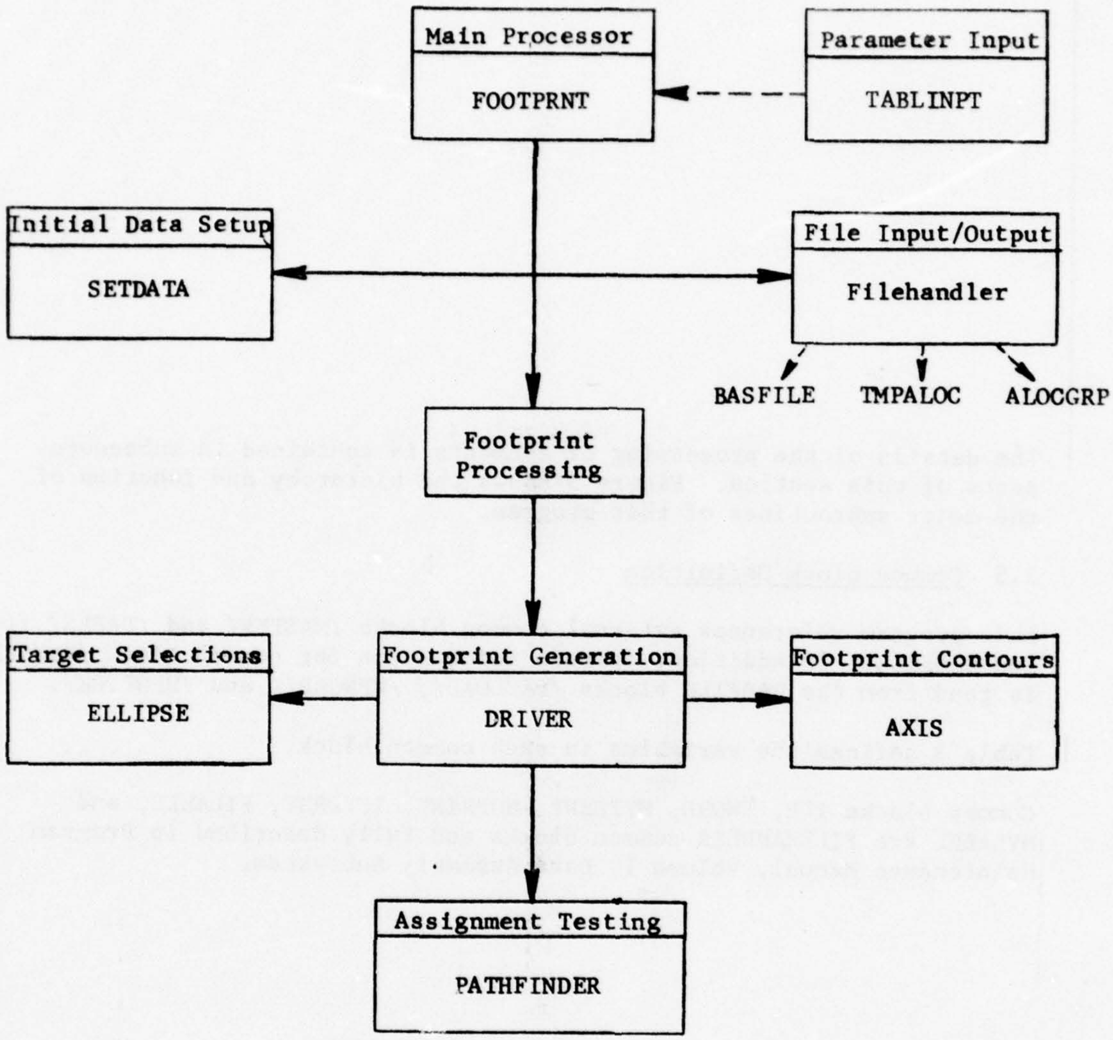


Figure 5. Hierarchy of Program FOOTPRNT Subroutine Execution

CONTENTS OF THIS PAGE INTENTIONALLY DELETED

CONTENTS OF THIS PAGE INTENTIONALLY DELETED

CONTENTS OF THIS PAGE INTENTIONALLY DELETED

Table 3. Program FOOTPRNT Common Blocks
(Part 1 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
AXIS	SMA(5)	Length of the semimajor axis for each of the five footprinting ellipses
	SMI(5)	Length of the semiminor axis for each of the five footprinting ellipses
BETA	BETA	Maximum testing angle for a target for inclusion in a footprint
CONTROL	NV	Number of boosters in the group
	NPASS	Processing pass number
CSAVE	FACT(16)	The fuel consumption rates for the particular first target being tested
CSTAT	CSTAT(7,4,2)	The values in common STATS for each internal pass and azimuth sweep
	LBOOST(2)	The last booster assigned for the first two internal passes
CSYS4	A0(16) A1(16) A2(16)	Fuel consumption parameters
	B0 B1 B2	Fuel load parameters
	RANGEB(2) BRADD(2)	Basic and azimuth dependent range coefficients for positive and negative launch aximuths
	CR2 CR1 CR0	Downrange-crossrange ratio parameters
	CRDEN	Distance scaling factor for the downrange-crossrange ratio
	UD2 UD1 UD0	Downrange-uprange ratio parameters
	LDSYS4	Length of the CSYS4 common block
DSQUARE	CD2	Square of the cross to down ratio
	UD2	Square of the up to down ratio
EARTH	RADIUS	Radius of the earth (nautical miles)
	PIDIV2	Pi/2
	DEGTORAD	Conversion factor for degrees to radians

Table 3. (Part 2 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
EDD	EDD(25,25)	The cost matrix for targets passed to PATHFIND
	MAXIN	The number of targets in the cost matrix
	RFIRST	The range to the first target
FILES	TGTFILE(2)*	Target data file
	BASFILE(2)*	Data base information file
	MSLTIME(2)*	Fixed missile timing file
	ALOCTAR(2)*	Weapon allocation by targets file
	TMPALOC(2)*	Temporary allocation file
	ALOCGRP(2)*	Allocation by group file
	STRKFIL(2)*	Strike file
	PLANTAPE**	Detailed plans tape
FOOTDATA	GAS	Fuel available for footprinting
	RX(2)	Range extension coefficient (positive and negative azimuths)
	RAX(2)	Azimuth dependent range extension coefficient
	TOSSC1(2,16) TOSSC2(2,16)	Fuel consumption parameters (for sign of azimuth and number of RVs still on board)
	TEONE(16) TETWO(16)	Fuel consumption exponents
	TDENOM(16)	Distance scaling factor
	RBASIC(2) RADD(2)	Basic and azimuth dependent range coefficients for positive and negative azimuths
	EONE ETWO	Downrange-crossrange ratio exponents
	DENOM	Distance scaling factor for downrange-crossrange ratio
	CONE(2) CTWO(2)	Downrange-crossrange ratio coefficients for positive and negative azimuths
	UE1 UE2	Downrange-uprange ratio exponents
	UC1(2) UC2(2)	Downrange-uprange ratio coefficients for positive and negative azimuths
	UDEN	Distance scaling factor for downrange-uprange ratio
	LLNGDAT	Length of the FOOTDATA common block

* First word is the logical unit number; second word is maximum file length in words. These files are all on disk.

**Logical unit number for the tape file.

Table 3. (Part 3 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
GRPDATA	R(1500)*	Distance from group centroid to DGZ (nautical miles)
	THETA(1500)*	Launch azimuth of weapon from group centroid to DGZ (radians)
	IFOR(1500)*	Linkage used in booster assignments
	IBOOST(500)	Index of first target assigned to booster
	LXLLFIX(42)	Fixed assignment indicator (1500 packed logical values)
HIT	HIT(16)	The index into EDD of the targets in the footprint
	NHIT	The number of targets hit
	SEPARATE	Logical value to determine if collocated targets can be hit sequentially
ISTART	ISTART	The internal index of the first target in the footprint
MASTER	IHDATE	Date of run
	IDENTNO	Run identification number
	ISIDE	Side
	NRTPT	Number of route points
	NCORRX	Number of corridors
	NDPEN	Number of depenetration points
	NRECOVER	Number of recovery points
	NREF	Number of refueling areas
	NREG	Number of regions
	NTYPE	Number of weapon types
	NGROUP	Number of weapon groups
	NTOTBASE	Number of weapon bases
	NPAYLOAD	Number of types of payloads
	NASMTYPE	Number of types of ASMs
	NWHDTYPE	Number of types of warheads
	NTANKBAS	Number of bases for tankers
	NCOMPLEX	Number of complex targets
	NCLASS	Number of weapon classes
	NALERT	Number of alert statuses
	NTGTS	Number of targets
	NCORTYPE	Number of types of corridors
	NCNTRY	Number of country codes

---

* Subroutine DRIVER equivalences arrays R, THETA, and IFOR to RANGE, AZM, and ISTAT.

Table 3. (Part 4 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
PARAMETER	MAXSYS	Maximum number of systems
	IHNAME(40)	Hollerith name of MIRV system
	NUMRVS(40)	The number of RVs carried by the bus
	MTYPE(40)	Functional form designator
PENADD	TOTFUEL	Total fuel available
	SRFC1 SRFC2	Spacing and release fuel coefficients
	SRFEXP1 SRFEXP2	Spacing and release fuel exponents
	SRFDEN	Distance scaling factor
	LPENDAT	Length of the PENADD common block
PRINT	PGRP(40)	Group for which optional prints are desired
	PSWP(40)	The azimuth sweep for optional prints
	PPAS(40)	The internal pass for optional prints
	PLIN(40)	The lowest internal index number for optional prints
	PHIN(40)	The highest internal index number for optional prints
	NP	The number of optional prints desired
RAIDATA	NT	Total number of strikes
	JGROUP	Group number
	JCORR	Corridor number
	INDEX(1500)	Target index number
	TGTLAT(1500)	Target latitude
	TGTLONG(1500)	Target longitude
	RVAL(1500)	Relative value of the target
	DLAT(1500)	DGZ offset latitude
	DLONG(1500)	DGZ offset longitude
	DESIG(1500)	Target designator code
	TASK(1500)	Target task/subtask code
	CNTRYLOC(1500)	Target country location code
	FLAG(1500)	Target flag code
	ISAL(1500)	Target salvo number
	NDEX(1500)	Array used by ORDER/REORDER
	LRAID	Length of common RAIDATA
	NTMAX	Maximum number of target assignments for one group
RAIDATA (as used in TRANFER)	NT(18003)	Scratch array used to transfer data
	LRAID	Length of common RAIDATA
	NTMAX	Maximum number of target assignments for one group

Table 3. (Part 5 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
RAIDATA	NT	Total number of strikes
(as used	JGROUP	Group number
is	JCORR	Corridor number
PATHFIND)	IPATH(33,25,16)	Index of closest city for every pass and branch
	SUBS(33,25)	Index number of initial city of subtour
	SUBL(33,25)	Length of the subtour
	NOLEG(33,25)	Packed array with forbidden leg information
	COST(33,25)	Cost of shortest path
	X(24,24)	Cost matrix for all points passed
	ST(15,24)	Subtotals for each stage and point
	IX(15,24)	Index of points
	LXBNP(34)	Logical flags for next branch and pass
	LXNBF(34)	Logical flags for branch and pass never branched from
	LXOPT(34)	Logical flags for optional branch and pass
	LXMCOST(34)	Logical flags for branch and pass with minimum cost
	LXNOSUBT(34)	Logical flags for branch and pass with no subtours
	EXTRA(36)	Buffer for RAIDATA to have constant size
RATIO	CDS	The downrange-crossrange ratio
	UDS	The downrange-uprange ratio
SHRTDATA	ALPHAZ(16)	
	ALPHA1(16)	Fuel consumption parameters
	ALPHA2(16)	
	BETAZ	
	BETA1	Fuel load parameters
	BETA2	
	MAXRBOST	Maximum booster range
	GTWO	
	GONE	Downrange-crossrange ratio parameters
	GZERO	
	DONE	Downrange-uprange parameters
	DZERO	
	LSHTDAT	Length of the SHRTDTA common block

Table 3. (Part 6 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
STATS	NHIT	Number of targets hit during current internal pass
	NFIX	Number of fixed targets hit during currant internal pass
	NBUS	Number of busses used
	NDUMP	Number of RV's dumped
	FOPT	Number of times PATHFIND failed to make a footprint because the optional path was too long
	FPB	Number of times the footprint failed after an excessive number of passes or branches
	FNOEL	Number of lead targets that had no legal ellipses
STRKSUM	KGROUP	Group number
	NTSTRK	Total number of strikes assigned
	NCORR	Number of penetration corridors used
	NSTRK(30)	Number of strikes assigned to each corridor
	LSTRKSUM	Length of common block STRKSUM
SYSMAX	ISYS	Index of MIRV system into PARAMETER arrays
	MAXBOOST	Number of boosters in the group
	MAXLOAD	RVs carried by bus
	MAXEDD	Maximum number of targets from which a footprint is built
	FUELOAD	Amount of fuel for the bus
	RMAX	Maximum range of the booster
TSCRATCH	ISCR	Logical unit number for assignment data scratch file
	ITABL	Logical unit number for footprint parameter data scratch file
WITHIN	FIT(5)	Logical flags indicating if a target is in each of the five ellipses for a specific first target
WPNGRPX	NWPNS(250)	Number of weapons in group
	NVEHGRP(250)	Number of vehicles in group
	WLAT(250)	Latitude of group centroid
	WLONG(250)	Longitude of group centroid
	ITYPE(250)	Weapon type
	IPAY(250)	Payload index
	ICLASS(100)	Class number

Table 3. (Part 7 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
	ISIMTYPE(100)	Hollerith name for type
	IMIRV(40)	MIRV system identification number

CONTENTS OF THESE PAGES INTENTIONALLY DELETED

CONTENTS OF THESE PAGES INTENTIONALLY DELETED

CONTENTS OF THESE PAGES INTENTIONALLY DELETED

2.6 Program FOOTPRNT

PURPOSE:	This is the main program. It acts as a control driver for the rest of the subroutines. It is the interface subroutine between this program and the remainder of the QUICK system.
ENTRY POINTS:	FOOTPRNT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CONTROL, CSTAT, DSQUARE, EARTH, FILABEL, FILES, GRPDATA, IFTPRNT, ITP, MASTER, MYIDENT, MYLABEL, PARAMETR, PRINT, RAIDATA, STATS, STRKSUM, SYSMAX, TSCRATCH, TWORD, WAROUT, SPNGRPX
SUBROUTINES CALLED:	DRIVER, GLOG, INITAPE, INITRANS, LREORDER, NEWCOOR, ORDER, RDARRAY, RDWORD, REORDER, SETDATA, SETREAD, SETWRITE, SKIP, SLOG, TABLINPT, TERMTAPE, TRANSFER, WRARRAY
CALLED BY:	Operating System; this is a main program.

Method:

The functioning of program FOOTPRNT can be divided into six parts; the flowchart and the following description are similarly divided. The parts are: the initialization of the program control variables, reading the strike data and determining the groups with the MIRV capability, setting the control data for each individual MIRV group, generating the footprints for each booster in the group and selecting, formatting, and writing the final plan, and finally terminating the processing. The majority of the file reading and writing is accomplished in this program and the specific cases are discussed in later paragraphs.

Part I - The Initialization of Control Variables

The functioning of this part of the program is quite straightforward logically. The program begins by calling subroutine INITAPE to initialize the filehandler. Subroutine TABLINPT is then called to read and interpret the user-input parameters. These parameters include the print requests, MIRV footprint parameter data tables. Then the majority of the basic weapon group information is read from the BASFILE. Commons /MASTER/ and /FILES/ are filled from the blocks of the same name on the BASFILE. Common /WPNGRPX/ is filed from BASFILE blocks /PAYLOAD/, /WPNGRP/, and /WPNT .E/. Finally, the variables EXTRAB and PEXTRA are read from block /E CESS/ on the BASFILE. This part finishes by initializing the TMPALOC and ALOCGRP files and requesting the preliminary prints.

Part II - Reading of the Strike Data and Determination of Groups with a MIRV Capability.

This section merely determines the data to be read from the TMPALOC file and places the data in core for processing by the remainder of the program. (For groups which do not have a MIRV capability, the data are copied from the TMPALOC file onto the ALOCGRP file for use by program POSTALOC.) This section begins by reading the data for common /STRKSUM/ from the TMPALOC file. This record contains the group number, the number of corridors for which strikes are planned, and the total number of strikes in each corridor. If the value of the group number is equal to an end-of-file marker (3HEOT) then the program goes to a termination block which finishes processing. (The termination block merely sets an end-of-file marker on the ALOCGRP file, terminates all the files, prints a termination message, and returns control to the system monitor.) If the end of processing has not been reached, the program determines the length of the fixed assignment indicator record. This record is a logical array which has been constructed by program ALOCOUT to show which weapons in the group were set by the fixed assignment capability of program ALOC. This indicator is used by later processors so that if various constraints require deletion of certain weapons, those weapons whose assignments were fixed by the user in program ALOC will not be among those deleted. Since the logical arrays are packed with 36 elements per computer word, the program must determine the length in words of this logical array which must be read for further processing. The program then determines if the current group has a MIRV capability. The first test is on the number of corridors in the strike data. If the number of corridors is greater than one, then the group must have bomber weapons since a weapon group with only missile weapons will send all its strikes to the same corridor, labeled 0. If the current group is a bomber group with more than one corridor for its strikes, the data are just copied out onto the ALOCGRP file and control returns to the beginning of this part to read the block of data for the next group. If only one corridor is assigned for the strikes of the group, the program reads the first three words of the /RAIDATA/ block. These words contain information on the number of the corridor to which these strikes are to be assigned. If the corridor number is not a 0, then again it is a bomber group and the data are merely copied out onto the ALOCGRP file. If the corridor number is a 0, it is a missile group. The next test checks the payload table to see if the attribute IMIRV has a value greater than zero for this group. If so, this missile group does have a MIRV capability and is a candidate for further processing. The final test is made before going on to the next part is to check for sufficient room in the arrays for the strikes assigned to this group. If there is not enough room, an error message is printed, the data are copied out onto the ALOCGRP file without further processing, and control is returned to the beginning of this part. If there is sufficient room to enable the later subroutines to process the data, control is transferred to Part III.

Part III - Setting Control Data for the Individual Group

Certain data must be preset before processing can begin on the MIRV group. Processing in this part is relatively straightforward. First, the Hollerith name of the weapon type is tested to see if there is agreement between a name input with the footprint parameter constraints and the name used by the QUICK system. If the names do not match, a warning message is printed and processing continues as usual. The program then calls subroutine SETDATA to retrieve the correct user's input footprint parameter.

Since the majority of data in the /RAIDATA/ block will be reordered and modified by later processing, it must be saved so that the correct data may be written out on the ALOCGRP. Therefore this part writes onto a scratch file, ISCR, the data that were read from the TMPALOC file into common /RAIDATA/. Table 5 displays the format of this scratch file. This part then calls subroutine NEWCOOR to convert the geographic information from latitude and longitude to range and launch azimuth from the group centroid. Finally, this part initializes all the arrays which make up potential target lists.

Part IV - Construction of Footprints and Select Final Plan

This part formulates two separate plans and then chooses the better one. In the first plan (a call to subroutine DRIVER with NPASS=1) the strikes are considered individually in order of increasing launch azimuth. In the second pass they are considered in order of decreasing azimuth. This method is used so that any potential target will be investigated by boosters whose initial assignment falls on either side of the launch azimuth of the potential target. Upon execution, subroutine DRIVER controls all processing in the detailed generation of assigning individual target strikes to specific boosters (or footprints). After each plan has been constructed program FOOTPRNT determines which is better and preserves that plan for print, further processing, and the eventual writing onto the ALOCGRP file. The plan deemed "better" is based on checks against the number of strikes assigned, number of fixed assignments honored, and the number of boosters generated for each plan. Comparisons are made against these three criteria in the order stated and the plan defined as better being the one where any one of the testing parameter exceeds the other. If all three parameters are equal, the second plan is chosen simply because it is already in working storage.

Part V - The Formatting, and Excess Booster Removal and the Final Plan for the Group

The later programs in the Sortie Generation subsystem expect the plan for missile groups to be in order of decreasing value of the booster assignments. That is, if we define the booster value to be the sum of the RVAL values of all the targets assigned to the booster, then the later processors expect these booster values to be decreasing as the lists are

CONTENTS OF THIS PAGE INTENTIONALLY DELETED

Table 5. Format for Assignment Data Scratch File

BLOCK	LENGTH	VARIABLE	DESCRIPTION
1	NT	INDEX	Target index number
2	NT	TGTLAT	Target latitude
3	NT	TGTLONG	Target longitude
4	NT	RVAL	Relative damage value
5	NT	DLAT	Offset latitude
6	NT	DLONG	Offset longitude
7	NT	NDEX	Sequence array for reordered data
8	NT	IFOR	Forward pointers for first pass
9	NV	IBOOST	First target for booster in first pass
10	NV	NTB	Number of targets assigned to booster in first pass
11	NT	NRVADD	Number of RVs added to each target in first pass

NT = Number of targets input from TMPALOC

NV = Number of boosters in group

output onto the tape. This part of program FOOTPRNT, therefore, computes the value of the strikes assigned to each booster. It then reorders the total target list so that the strikes are in order not only by booster value but by order of delivery by the MIRV equipment. This function is done by assigning to every booster a 'ranking' index which is defined as follows:

$$R = -(\Sigma(RVAL + FACT))$$

where R = ranking index

Σ = summation over all strikes assigned to a booster

RVAL = marginal damage value of individual assignment

FACT = 10,000 for normal assignment and the square if fixed assignment

Thus once each booster has been assigned a ranking index, subroutine ORDER is executed to order the boosters based on the defined parameter R. Ranking places most importance on boosters that contain fixed assignments followed by the number of individual strikes assigned to a given booster.

A determination is now made of the actual number of boosters that are to be assigned in this group. Program PREPALOC added (if user exercised) some extra reentry vehicles so that the allocator would assign an excess of targets to the group. These extra reentry vehicles were added to aid the processing in program FOOTPRNT. Program ALOC does not consider footprint constraints when allocating weapons (strikes) to targets. It is possible, therefore, that some of the targets assigned by the allocator cannot be put into a feasible footprint. Therefore, the excess of weapons allows for generating a total assignment which when constraints are applied does not produce an underutilization of weapons. Thus, this part of the program now must determine the actual number of boosters (hence to individual weapons) as data base defined (parameter NVEHGRP for weapon group JGROUP). Simply, the least valuable (in terms of ranking index R) are deleted from the selected plan and corresponding counts adjusted.

Salvo numbers are assigned sequentially to each reentry vehicle. Assignments are made between the minimum, MINSAL, and maximum MAXSAL, salvo number. For each salvo, NSIM launches are made.

Standard prints for each group processed are now produced. Following that a sequencing array (IR) is defined whose use will be to resort the individual strikes in the proper sort order for final output. This process is conducted within DO loops up to statements 4100 and 4110. Array NDEX, contains the sorted order of each booster; array IBOOST points to the first target within the booster; array IFOR points to the next target if any exists. By properly queuing IBOOST, then IFOR, array IR may

be easily stated. Upon final definitions of array IR, individual strike associated parameters are reordered and data written onto the ALOCGRP file and control returns to the beginning of Part II where the strike data for the next group are read.

Part VI - Termination Block

Processing has been concluded; all files are terminated and end messages printed.

Program FOOTPRNT is illustrated in figures 6 and 7.

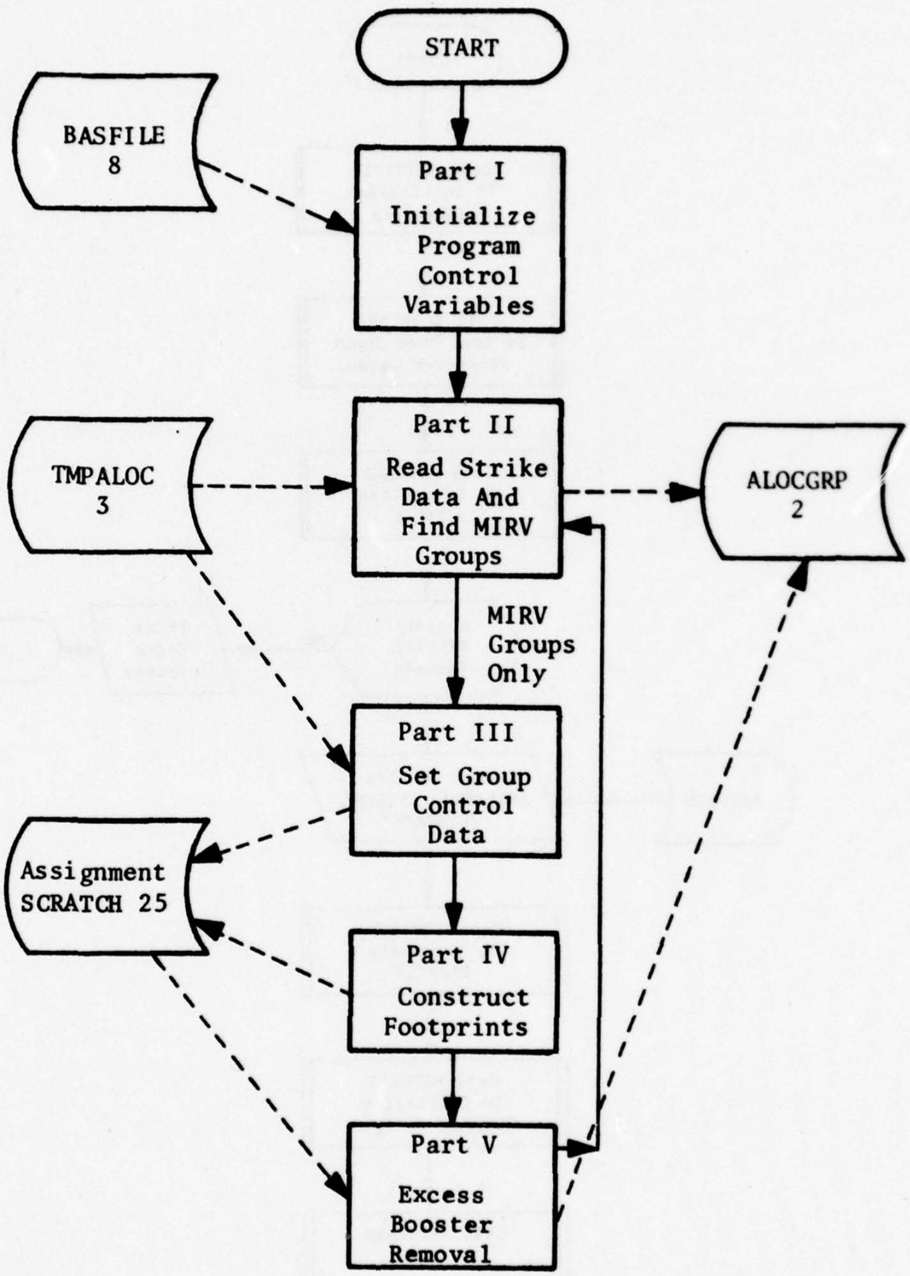


Figure 6. Program FOOTPRNT (General Flow)

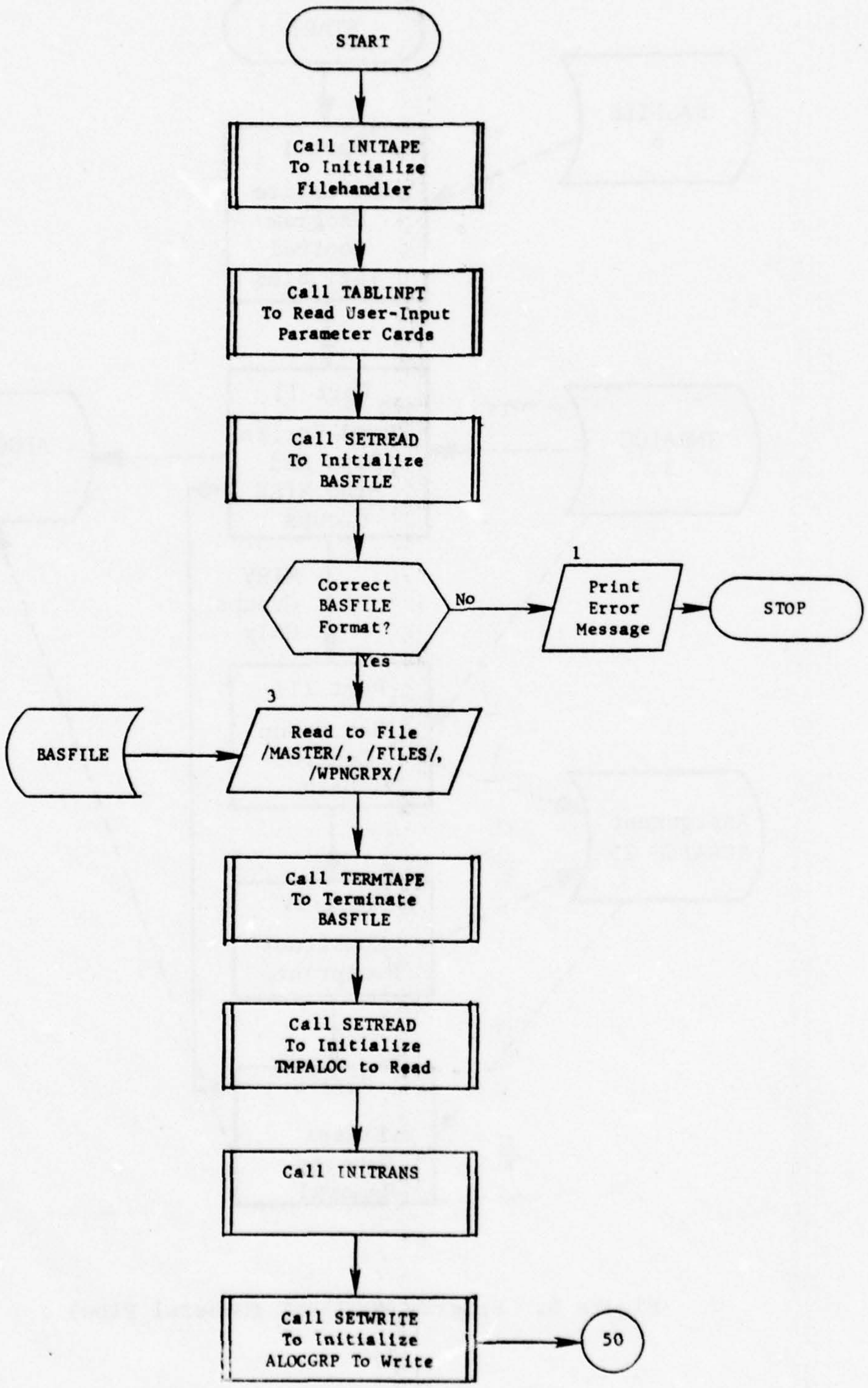


Figure 7. Program FOOTPRNT (Detailed Flow) (Part 1 of 6)
Part I: Control Variable Initiation

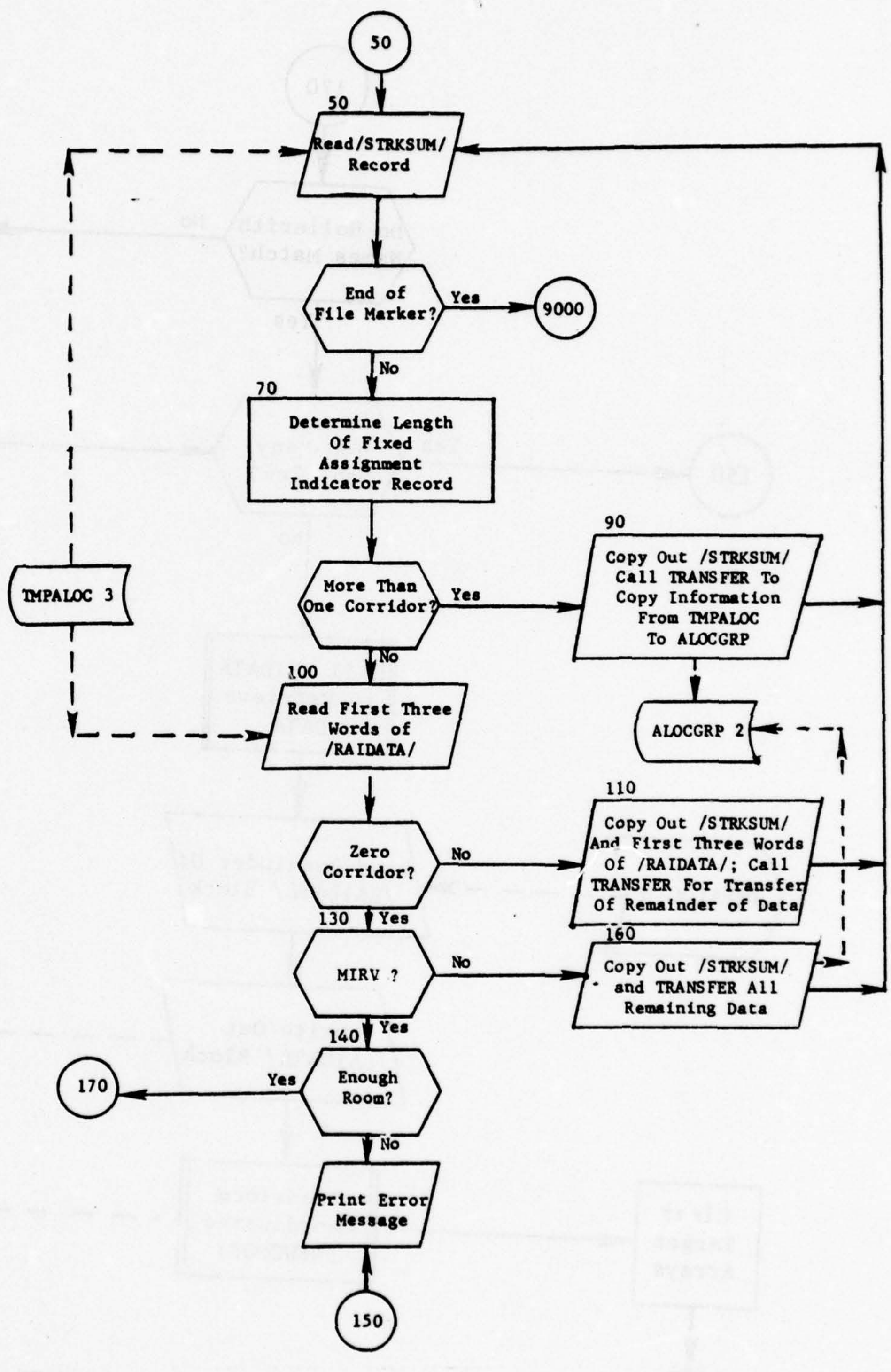


Figure 7. (Part 2 of 6)
Part II: Read Strike Data and Find MIRV Groups

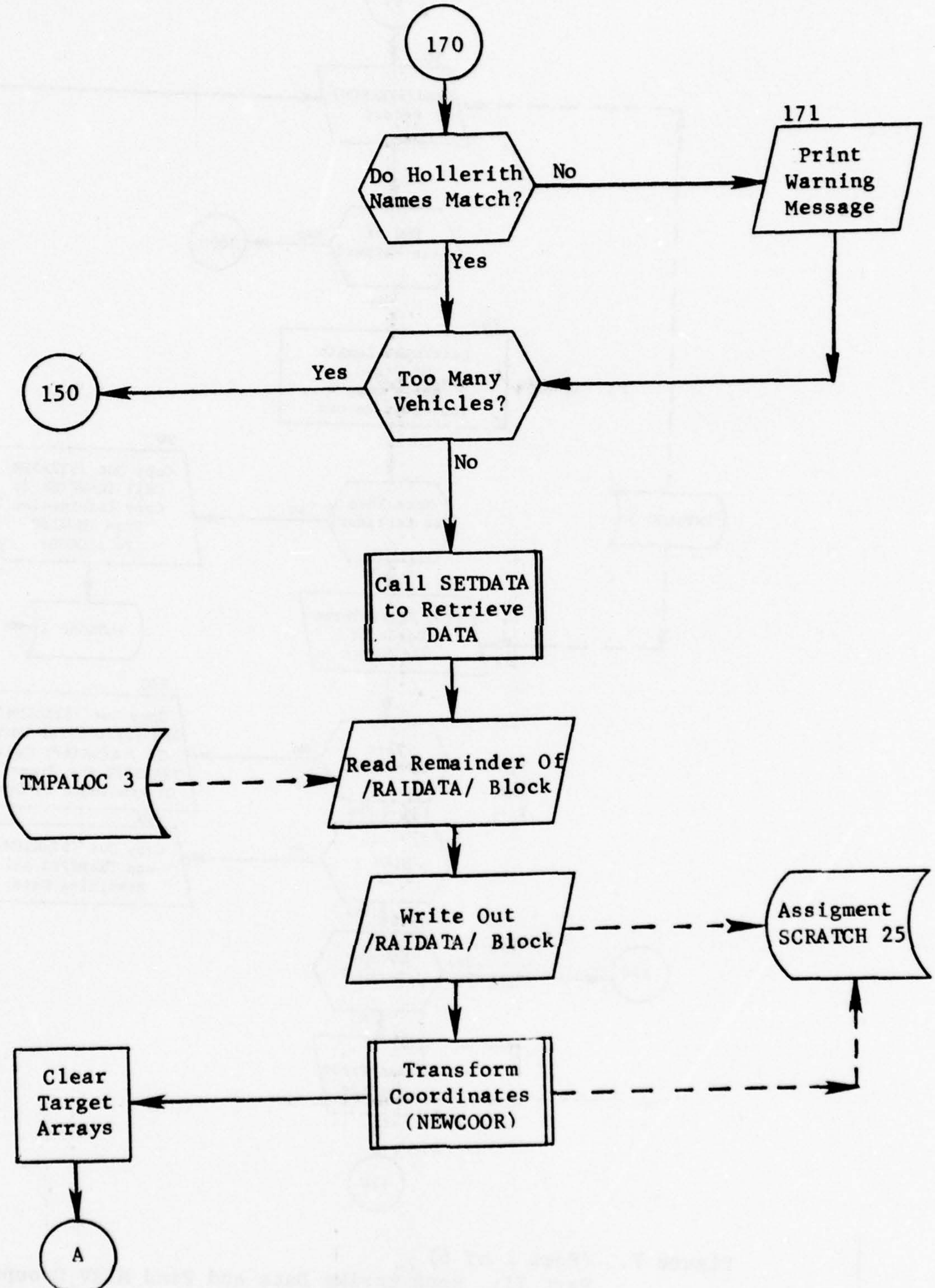


Figure 7. (Part 3 of 6)
Part III: Set Group Control Data

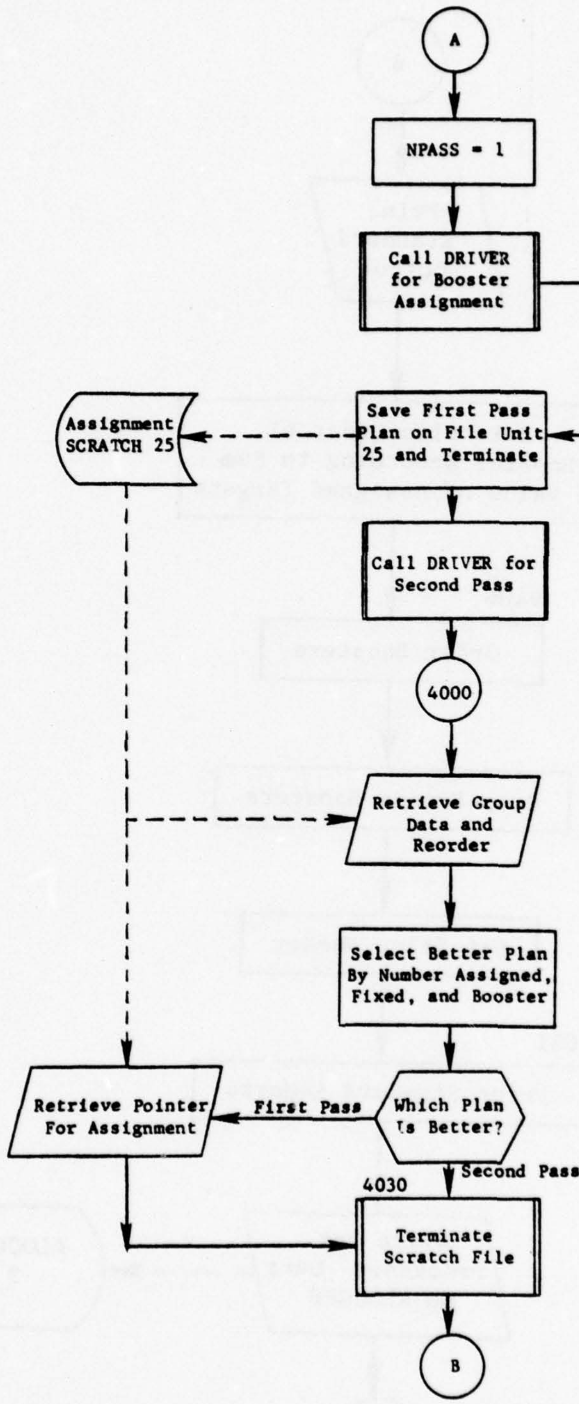


Figure 7. (Part 4 of 6)
Part IV: Construct Footprints and Select Final Plan

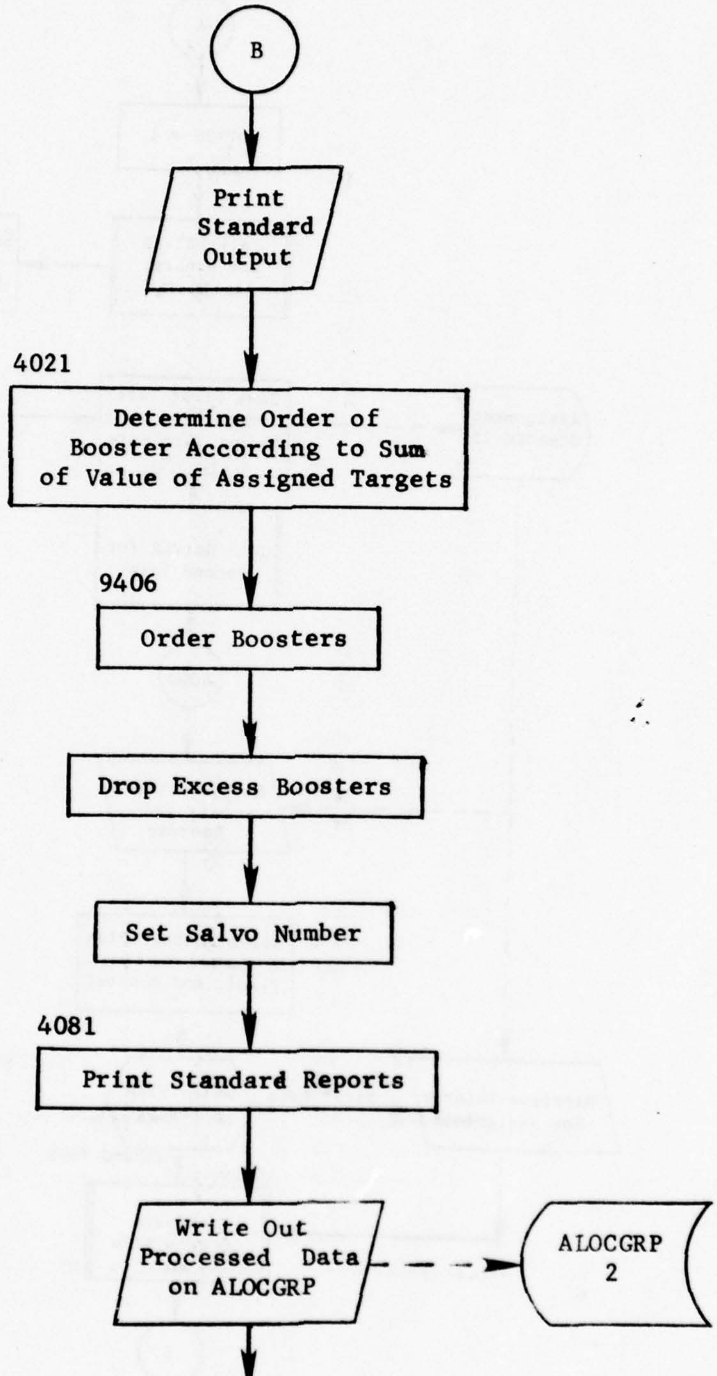


Figure 7. (Part 5 of 6)
Part V: Formatting and Excess Booster Removal (Detail)

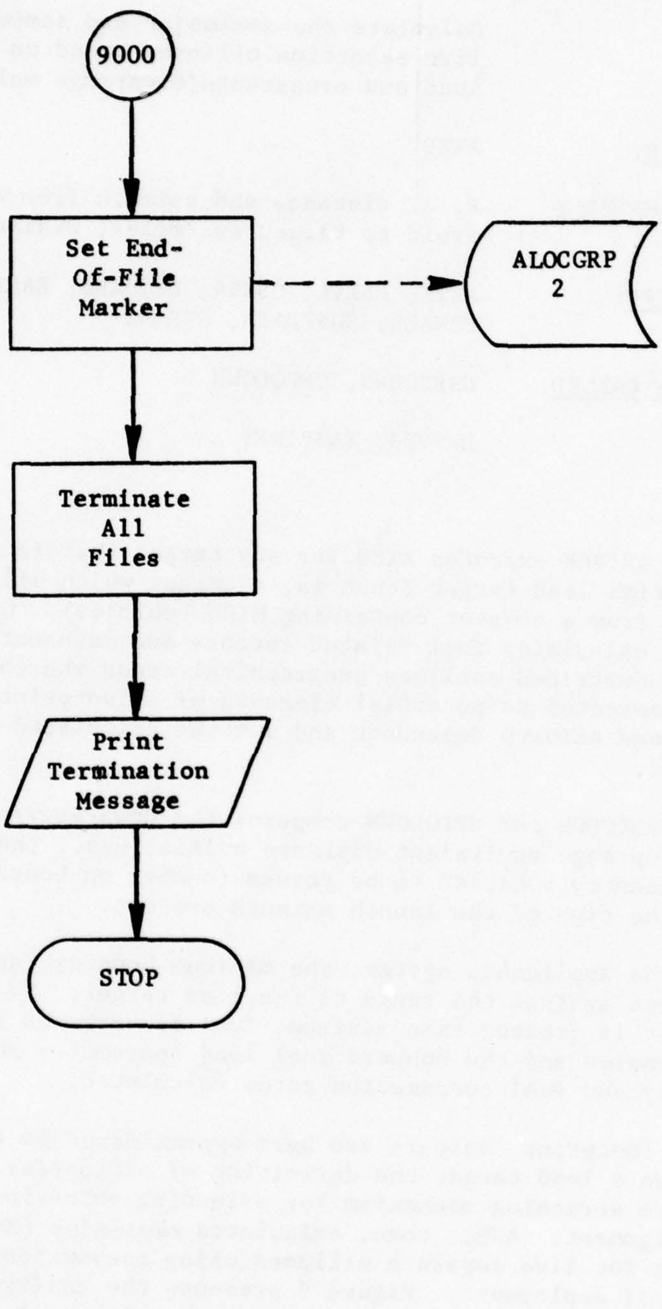


Figure 7. (Part 6 of 6)
Part VI: Termination Block

2.6.1 Subroutine AXES

PURPOSE: Calculate the semimajor and semiminor axes of five selection ellipses based on calculated fuel load and crossrange/downrange multipliers.

ENTRY POINTS: AXES

FORMAL PARAMETERS: R, AZ distance and azimuth from weapon group centroid to target (n. miles, radians)

COMMON BLOCKS: AXIS, CSAVE, CSYS4, DSQUARE, EARTH, FOOTDATA, PENADD, SHRTDATA, SYSMAX

SUBROUTINES CALLED: CRSTODWN, UPTODOWN

CALLED BY: DRIVER, TABLINPT

Method:

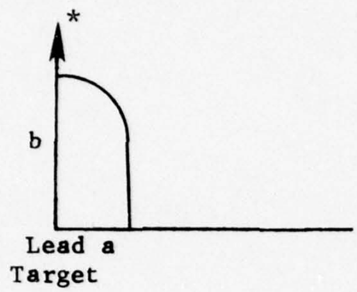
Subroutine DRIVER executes AXES for any target that is being considered as a potential lead target (that is, a target which will be the first assignment from a booster containing MIRV vehicles). Given this assumption, AXES calculates fuel related factors and mathematical coutours which when described outlines geographical areas whereby other targets are to be selected as potential elements of a footprint. The contours are range and aziumth dependent and must be calculated anew for each lead target.

Function CRSTODWN and UPTODOWN computes the downrange/crossrange and downrange/uprange equivalent distance multipliers. The maximum load (ML1) of reentry vehicles to be tossed (number on board at launch minus one) and the sign of the launch azimuth are set.

Based on the applicable system, the maximum booster range is calculated and compared against the range to the lead target. If the range to the lead target is greater than maximum, fuel is borrowed from the bus for range extension and the onboard fuel load (parameter FUELOAD) decremented accordingly and fuel consumption rates calculated.

Potential footprint contours are best approximated by an ellipse. Therefore, given a lead target the definition of ellipse(s) serves as an appropriate screening mechanism for selecting potential targets for footprint assignment. AXES, then, calculates semimajor (SMA) and semiminor (SMI) axes for five separate ellipses using assumptions based on the extremeties of deployment. Figure 8 presents the configuration for ellipses labelled 1, 3, and 5. Ellipse 2 equals ellipse 1 with a clockwise rotation of 45°; ellipse 4 equals ellipse 5 with a counterclockwise rotation of 45°. Targets to be considered for assignment must be at least geographically located within the defined ellipses for a given lead target. Upon selection of the proper number of targets within the ellipses they will be further tested for feasibility.

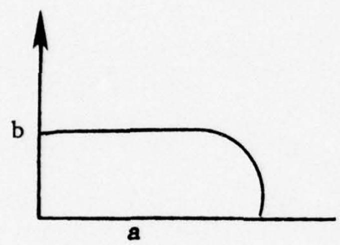
Figure 9 illustrates subroutines AXES.



Ellipse 1

b = Maximum distance with one on-board vehicle that can be thrown downrange

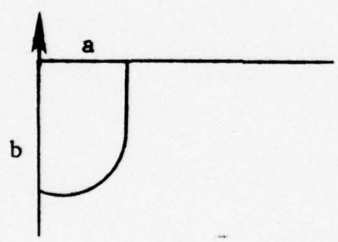
a = Maximum distance with ML1** on-board vehicles that can be thrown crossrange



Ellipse 3

b = Maximum distance with ML1 on-board vehicles that can be thrown downrange

a = Maximum distance with one on-board vehicle that can be thrown crossrange



Ellipse 5

b = Maximum distance with one on-board vehicle that can be thrown uprange

a = Maximum distance with ML1 on-board vehicles that can be thrown crossrange

---

* Arrow indicates downrange direction

**ML1 equals initial loading factor minus one

Figure 8. Maximum Footprint Ellipses Related to Lead Target

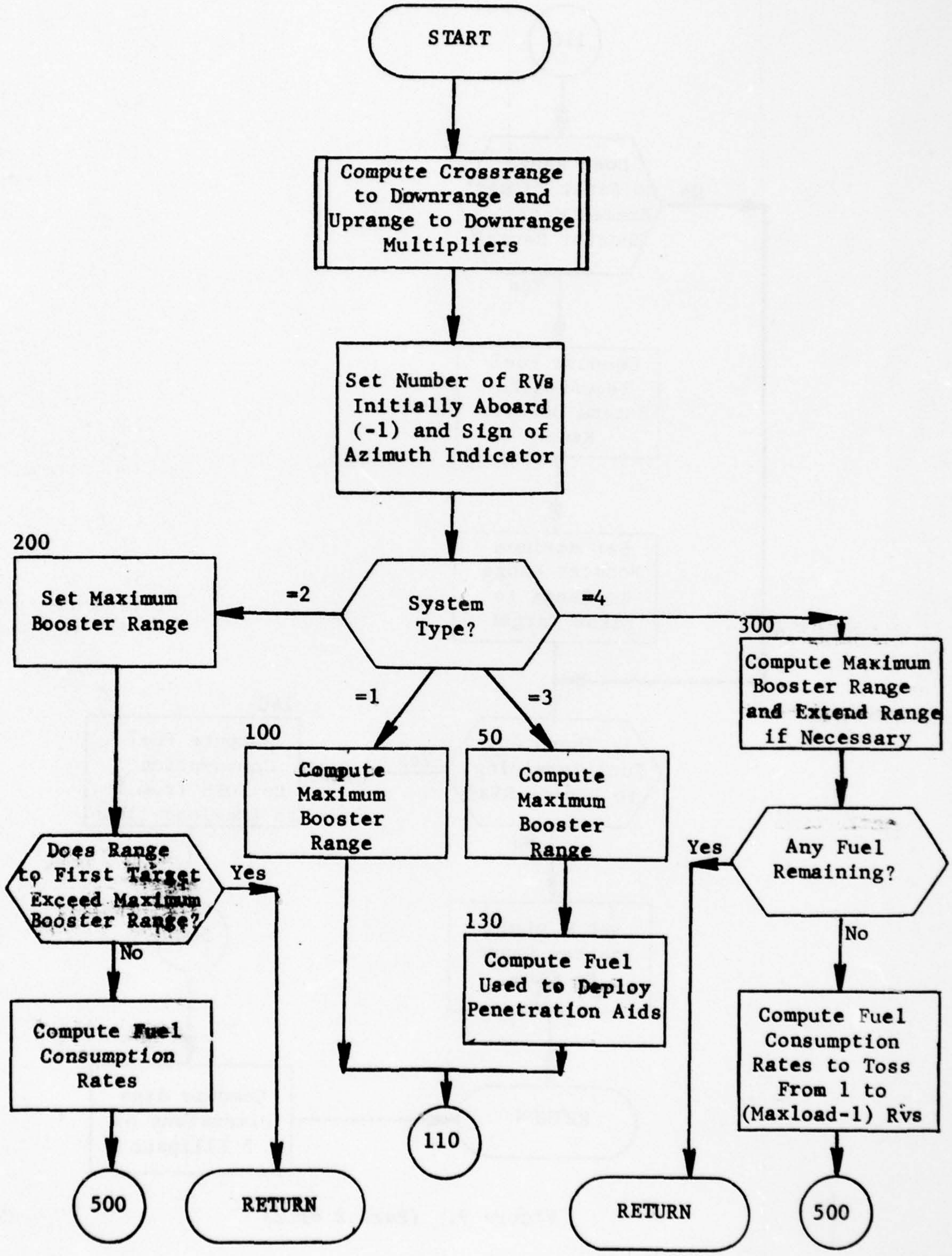


Figure 9. Subroutine AXES (Part 1 of 2) CH-3

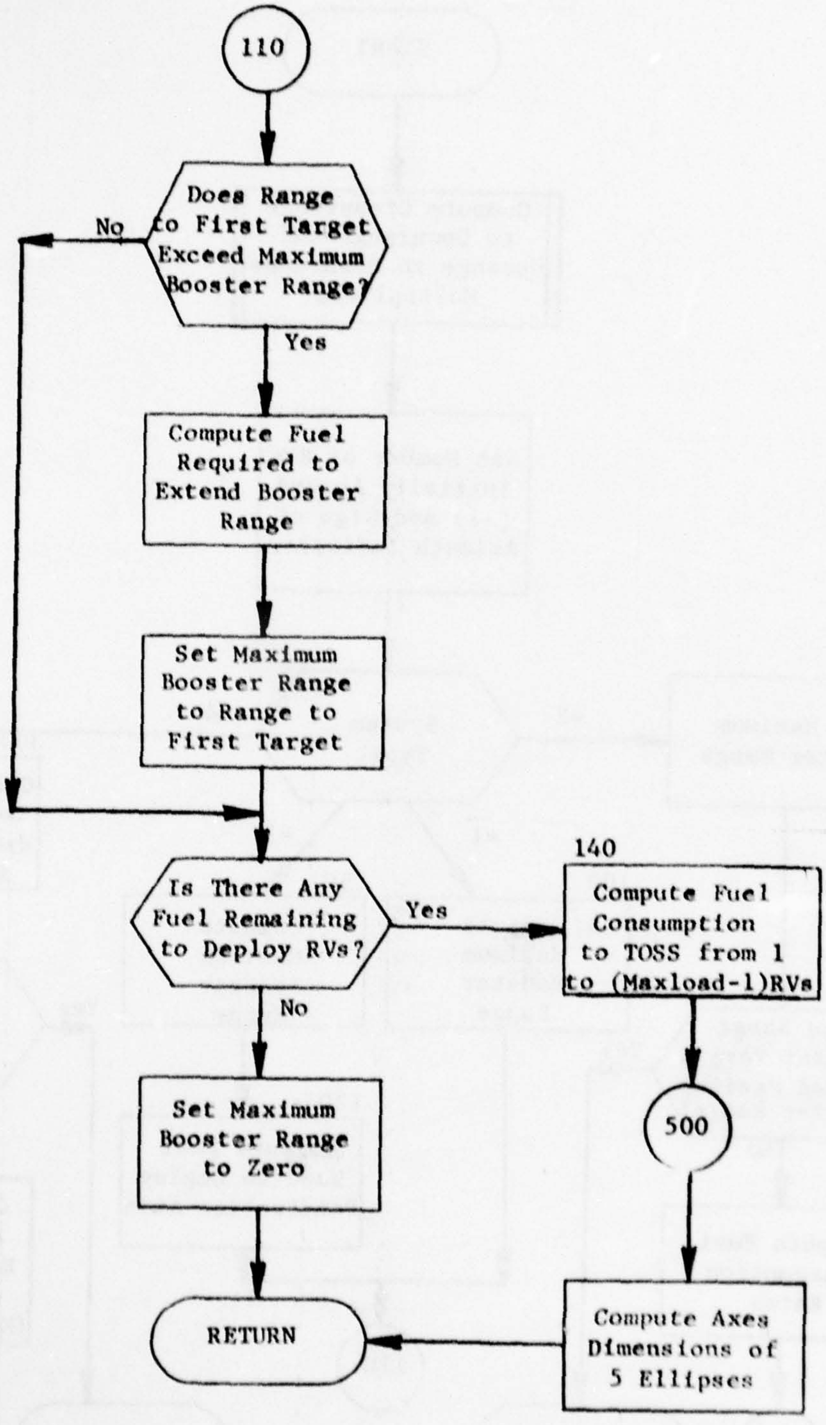


Figure 9. (Part 2 of 2)

CH-3

2.6.2 Function CRSTODWN

PURPOSE: This routine calculates the crossrange-downrange ratios.

ENTRY POINTS: CRSTODWN

FORMAL PARAMETERS:
I - System type index
R Range to first target (nautical miles)
AZ- Launch azimuth to first target

COMMON BLOCKS: CSYS4, FOOTDATA, PENADD, SHRTDATA

SUBROUTINES CALLED: None

CALLED BY: AXES

Method:

This function simply applies the crossrange-downrange ratio equations whose parameters were input in subroutine TABLINPT. The formal parameters are the system type index, the range and azimuth to the first target.

Function CRSTODWN is illustrated in figure 10.

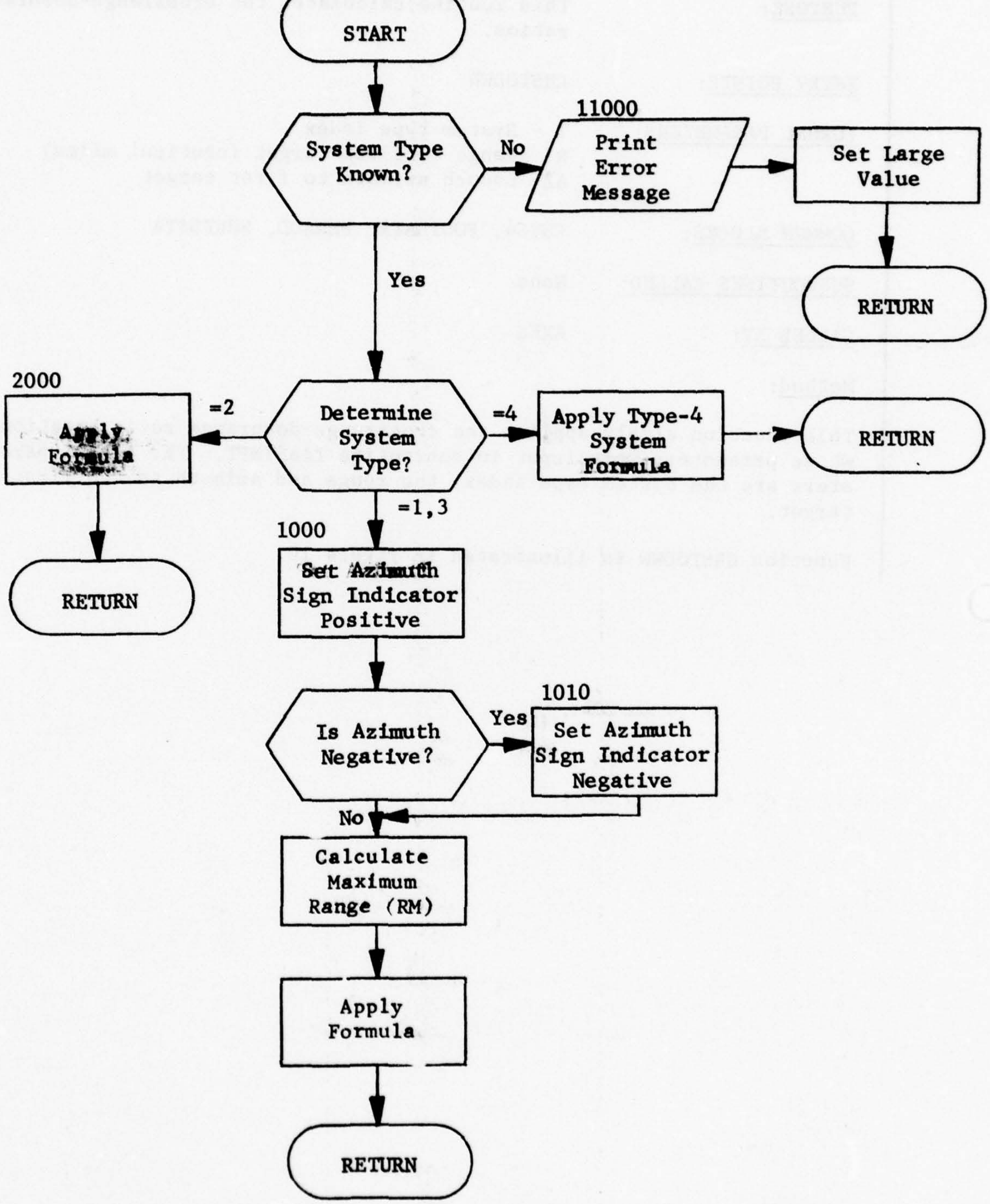


Figure 10. Function CRSTODWN

2.6.3 Subroutine DRIVER

PURPOSE:	Individual strikes are assigned to specific boosters.
ENTRY POINTS:	DRIVER
FORMAL PARAMETERS:	None
COMMON BLOCKS:	AXIS, BETA, CONTROL, CSTAT, DSQUARE, EARTH, EDD, GRPDATA, HIT, ICOUNT, IEDD, IEDDC, IEDDF, ISTART, PRINT, RAIDATA, RATIO, STATS, SYSMAX, WAROUT, WITHIN
SUBROUTINES CALLED:	AXES, ELLIPSE, GLOG, PATHFIND
CALLED BY:	FOOTPRNT

Method:

DRIVER is executed twice for each weapon group that contains MIRV systems. For each call (referred to as passes) individual strikes are assigned to specific boosters. The first call, Pass 1, investigates the target list ordered by increasing azimuth; the second call, Pass 2, investigates the target list in the reverse order from the first call. FOOTPRNT determines the better of the two plans.

Prior to detailed explanation of subroutine DRIVER, certain usages of terminology should be clarified and are:

- Lead target -- The first target assigned within a footprint. Fuel factors are computed relative to this target.
- Fixed target -- A target assignment directed by the user.
- Dumped target -- Any target that has a strike generated within FOOTPRNT assigned to it. The end result is the hitting of a target more than once.
- Booster or Bus -- The platform that carries the MIRV's.
- Sweep -- Investigation of the entire target list based on precise assumptions
- Ellipse -- Geographic contour relative to a lead target. For a feasible footprint, potential targets must be located within the boundaries of the ellipse(s). However, targets located within an ellipse are not necessarily feasible footprintable targets.

- o Cone -- Method of target selection. Neighboring target on the immediate right or left (in the azimuth sense) of a lead target are selected.

Major controlling parameters that directs the logic flow consists of:

- o ISTAT -- Status array that indexes into the input target list. Initially set to zero and reset to the next target or a booster number whenever a target is assigned to a feasible footprint.
- o NPASS -- Pass indicator. Equals 1 for search on increasing azimuth; equals 2 for search on decreasing azimuth.
- o INPASS -- Sweep indicator. Equals 1 for fixed assignment sweep; equals 2 for normal sweeps, equals 3 for dumping sweeps.
- o ISTART -- Target index (into the ISTAT array) that is being considered as a lead target for a given booster.
- o IEDDF -- Array that contains a list of indexes into the ISTAT array that are to be tested for feasible footprint determination. Parameter MAXIN defines the number of entries.
- o HIT -- If a footprint is feasible (from the IEDDF list), the target IEDDF indexes are placed in the HIT array in the order of assignment.
- o EDD -- Equivalent downrange distance array of going from target I to target J.
- o SEPARATE -- Set TRUE if calculated targets are not permitted within a footprint; set FALSE if located targets are permitted.

General

DRIVER's major modes of operation consists of looping through the entire target list (in the correct sort) and defining a target as being lead and from that assumption collecting a subset of targets that potentially may form a footprint. These targets are passed to subroutine PATHFINDER to determine if a feasible footprint exists within the subset. If a footprint does exist, the individual strikes are assigned to the booster being formed; if not, the next target is considered as lead and the processing continues until the entire target list has been processed.

For each execution of DRIVER, the entire target list is 'swept' three ways with separate assumptions made on each sweep. Commentary to follow presents the assumptions made for each sweep followed by methods of

target selection common to each sweep and, finally, target assignment is discussed.

Sweep 1 (INPASS=1)

Only user fixed assignments are considered for lead target consideration. However, companion potential targets need not be fixed; only the lead target. This method of selection places a high premium in guaranteeing the assignment of fixed targets. Selection of targets consists purely of the ellipse algorithm; no other method is used for this sweep.

Sweep 2

This sweep is referred to as the normal sweep; that is there is no restriction on the targets to be selected. For this sweep, the target list is gueried four times based on two settings of parameter SEPARATE (see above for definition). For each setting of SEPARATE, potential targets are collected that are geographically located within defined ellipses from a lead target and following that query, the entire target list is again queried collecting targets that neighbors a lead target on both sides (left or right). The first query is termed the ellipse method of selection, the second, the cone method. Details of both are discussed later.

All possible attempts of forming footprints whereby a given footprint does not 'hit' a single target more than once are considered. However, there are many data base scenerios where a footprint may be formed only through the consideration of duplicate assignments. In particular, this situation exists for a weapon group assignment where many strikes are assigned to only one or two individual targets. The existence of this fact, and other scenerios, dictates the relaxing of the duplication restriction. SEPARATE communicates with the assignment subroutine (PATHFINDER) as to the level of restriction.

Sweep 3

Each booster within the weapon group (of which there are NV in number) must completely utilize its onboard payload inventory. Say, for example, a booster may toss three reentry vehicles; each footprint, then, must have three strikes. A total of one or two strikes per booster is not permitted. This sweep (called the dumped sweep) is necessary only if the correct number of footprints were not formed due to target spacing. For this sweep, the number of targets needed to define a footprint will be sequentially reduced by one unit for a sweep of the entire target list and the targets being considered as a lead are assigned multiple strikes, or are dumped on, by an amount equalling the reduction. Defining parameter MAXLOAD as the inventory payload number, there may be up to MAXLOAD queries through the target list. For each reduction only the cone method of target selection is employed and parameter SEPARATE set to consider coplanor targets.

Target Selection

For any sweep, the ISTAT array is queried for potential target selection. If an entry into ISTAT equals zero that target is unassigned; if the entry is not zero it has been assigned and need not be considered anew. Upon finding an entry of zero into ISTAT, that target is called lead and ISTAT is further investigated for selection potential targets to be footprinted. Both the ellipse and the cone method selection are discussed below.

For each lead target, subroutine AXIS is called to calculate the dimensions of five separate ellipses relative to a lead target. Now, for each entry of zero into ISTAT, subroutine ELLIPSE determines if the considered target is geographically located within the ellipses. If so, array ICOUNT and IEDD are updated and the remaining targets are processed. This continues until a cutoff filter is exceeded (azimuth BCRIT) or until one ellipse collects a proper amount of potential targets (parameter MAXEDD which equals the onboard payload inventory plus 50 percent).

Upon meeting the BCRIT cutoff filter, all five ellipses are counted to find the one with the largest number of entries which must at least equal the payload inventory (MAXLOAD) for footprint consideration. If no single ellipse contains a footprintable number of entries, the counts from two ellipses are merged into one. The merging expands the geographic areas by considering neighboring ellipses that overlap each other to a large degree. Five new counts are now formed and the one with the maximum number of entries is considered for footprinting. If the merged counts fail to contain at least MAXLOAD entries, the entire collection of potential targets is formed into one list and that list is then considered for footprinting.

If the selection process produces a feasible footprint, arrays are updated and a new lead target is chosen. If a selection criteria failed to produce a footprint, the lead target in question is not dropped until all possible selection combinations have been tested. That is, if one individual collection failed to produce a footprint, all other possible selections will be attempted. If the ellipse 1 collection failed, the ellipse 3 collection may produce a footprint, for example.

The ellipse collection always 'looks' in one azimuth direction relative to a lead target. This prevents a snaking effect and also makes best use of crossrange fuel consumption rates. However, certain target collections may only be properly assigned upon permitting the searching of all azimuths relative to a lead target. This method of target selection has been referred to as the cone selection.

The algorithm for cone selection attempts to pick targets such that the azimuth of a lead target centers the final collection. First, the nearest target to the right of a lead target is selected. Second, all targets to the left of a lead target whose azimuth (relative to lead) is less than or equals the azimuth of the chosen target to the right of a

lead are selected. Third, an additional target to the left is chosen. Now, processing reverts to the right of a lead target and all targets selected up to the azimuth equalling the azimuth of the last target selected on the left of a lead target. One more target is chosen and processing continues to the method outlined. Processing for a given lead target terminates upon collecting MAXEDD entries or exhaustion of the complete target list.

Target Assignment

Array IEDDF contains a list of target indexes that are to be considered for footprinting. From this array, the Equivalent Downrange/Distance array, EDD, is computed which contains the costs (distances) of going from target I to target J. Subroutine PATHFINDER interrogates the array and formulates the optimal footprint, if it exists. PATHFINDER returns parameter NHIT and array HIT. If NHIT equals zero, no footprint could be formed. If NHIT is positive it equals the number of entries in the footprint. Array HIT contains the indexes into the IEDDF array of the target that are assigned to the footprint and the order of assignment.

Subroutine DRIVER is illustrated in figure 11.

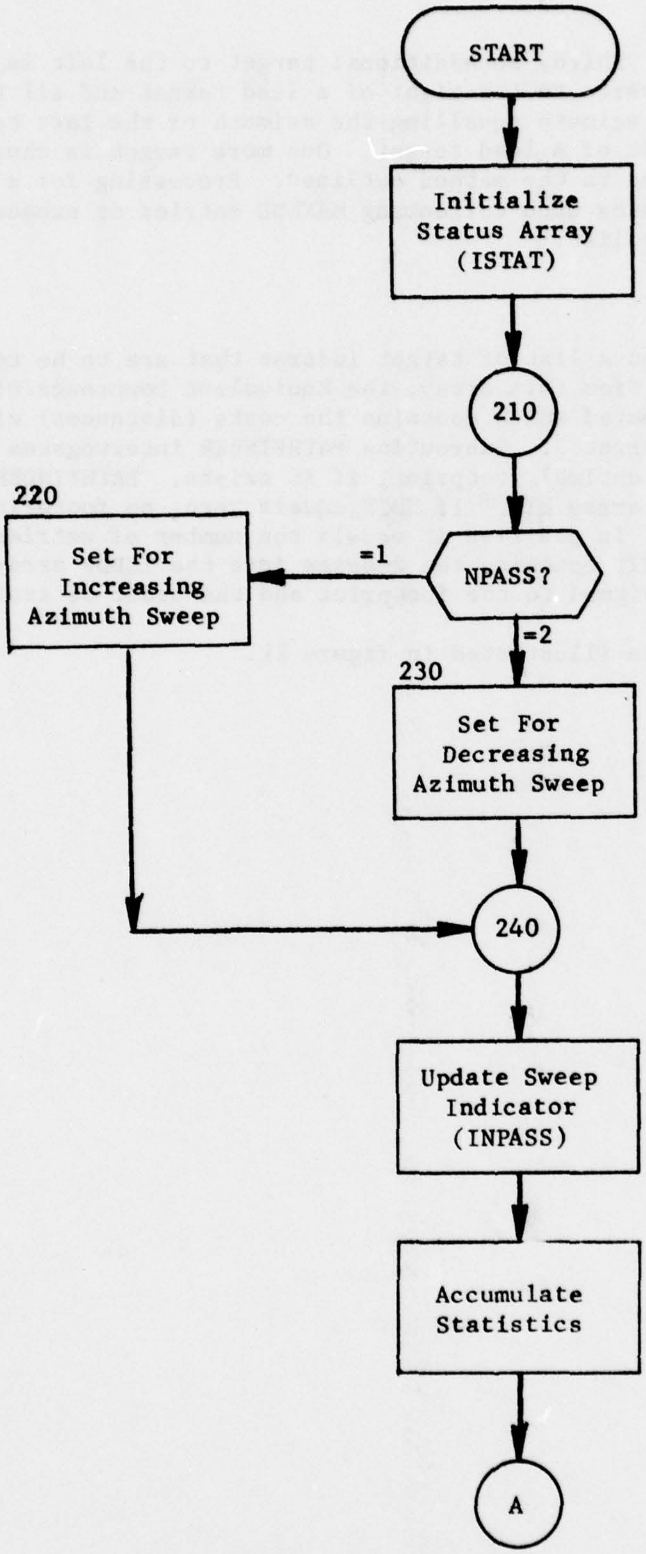


Figure 11. Subroutine DRIVER (Part 1 of 8)

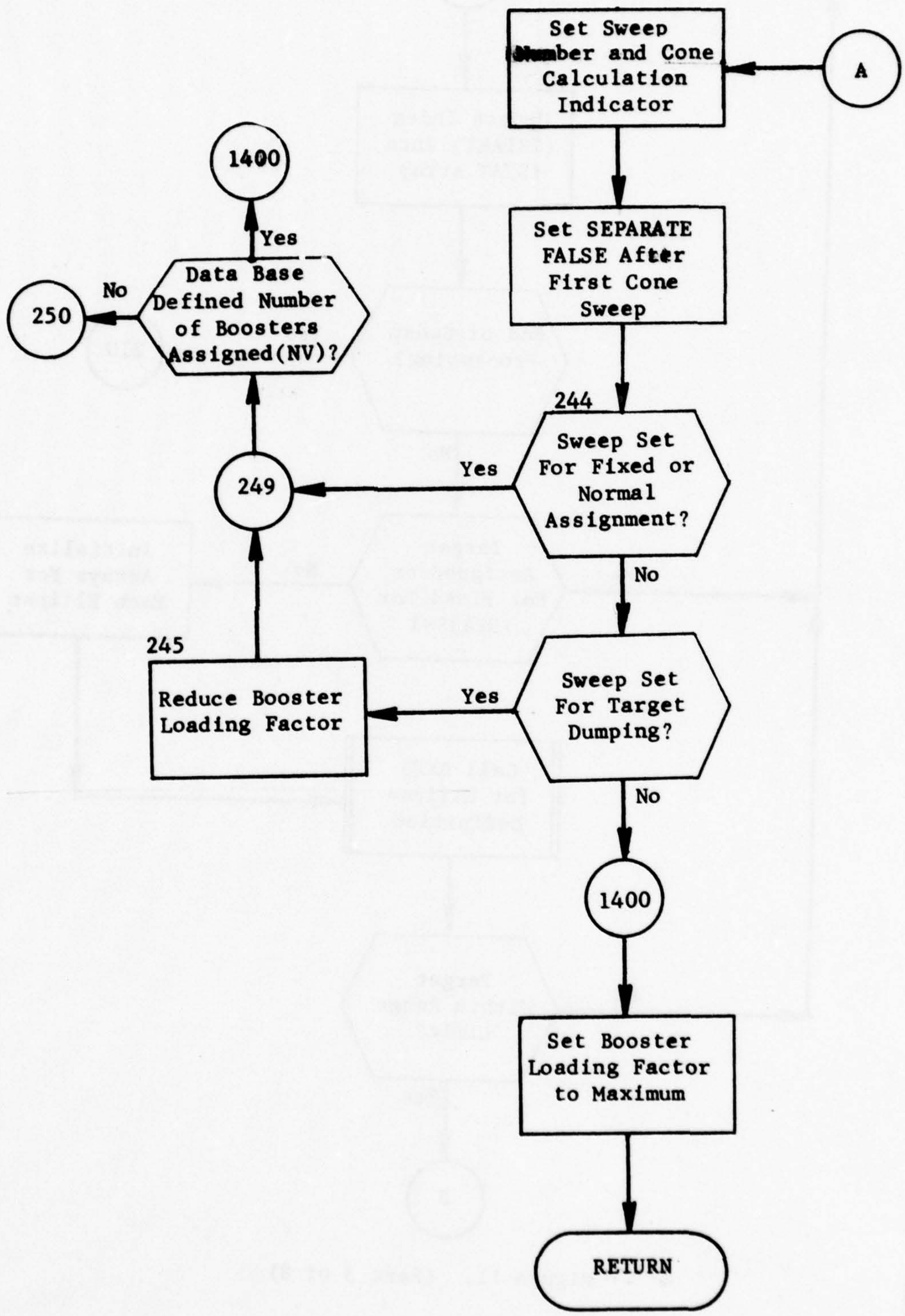


Figure 11. (Part 2 of 8)

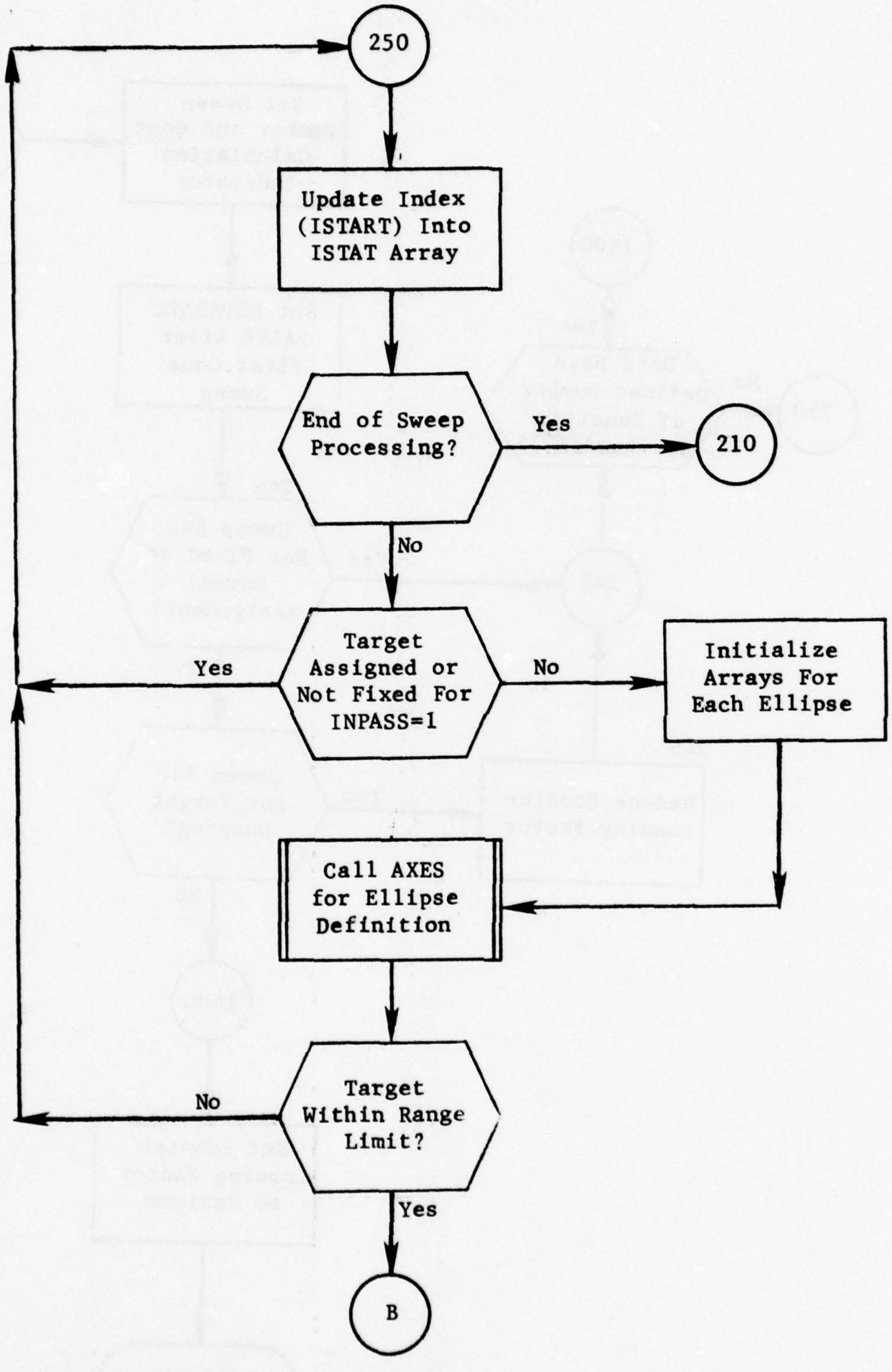


Figure 11. (Part 3 of 8)

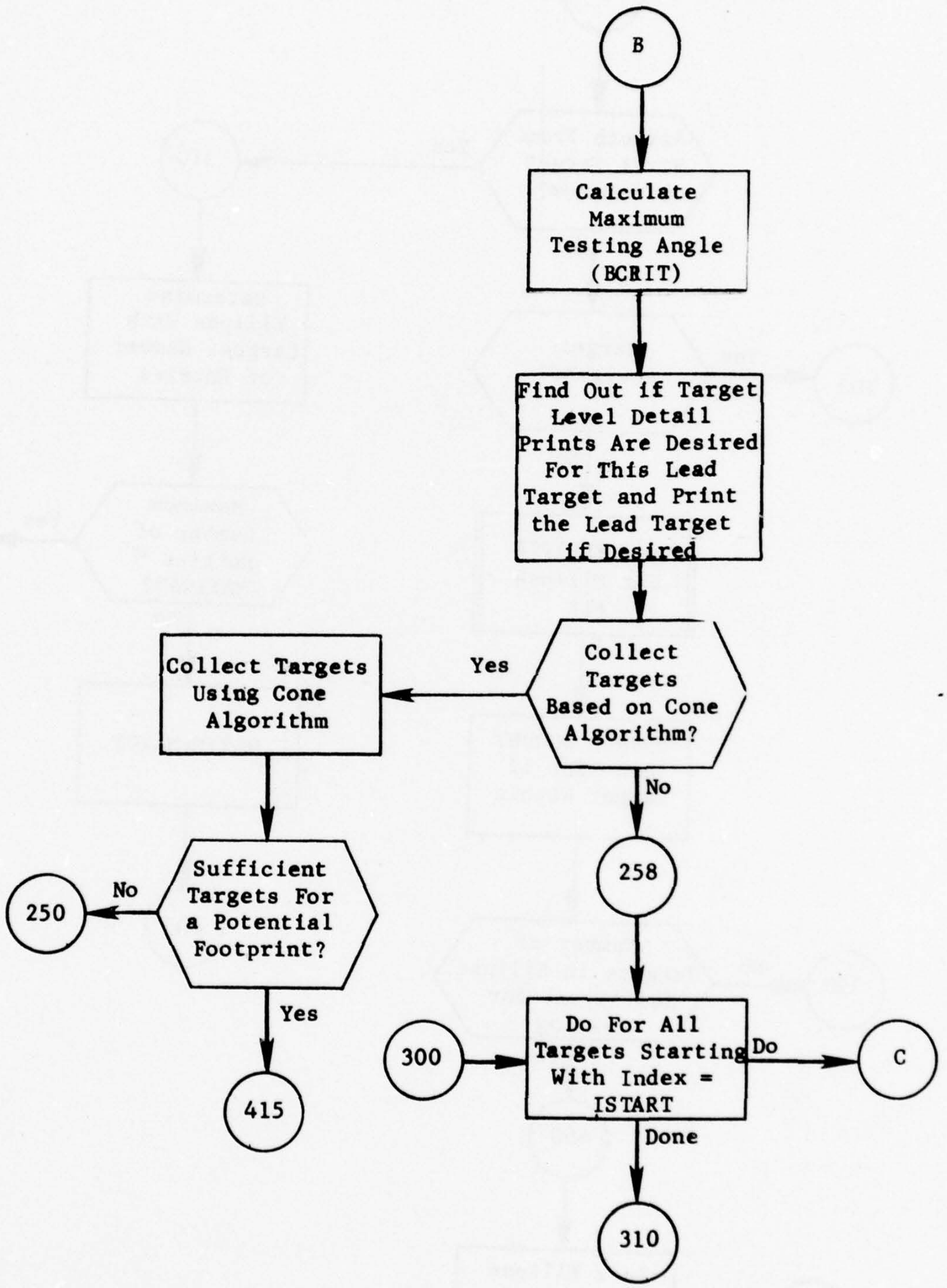


Figure 11. (Part 4 of 8)

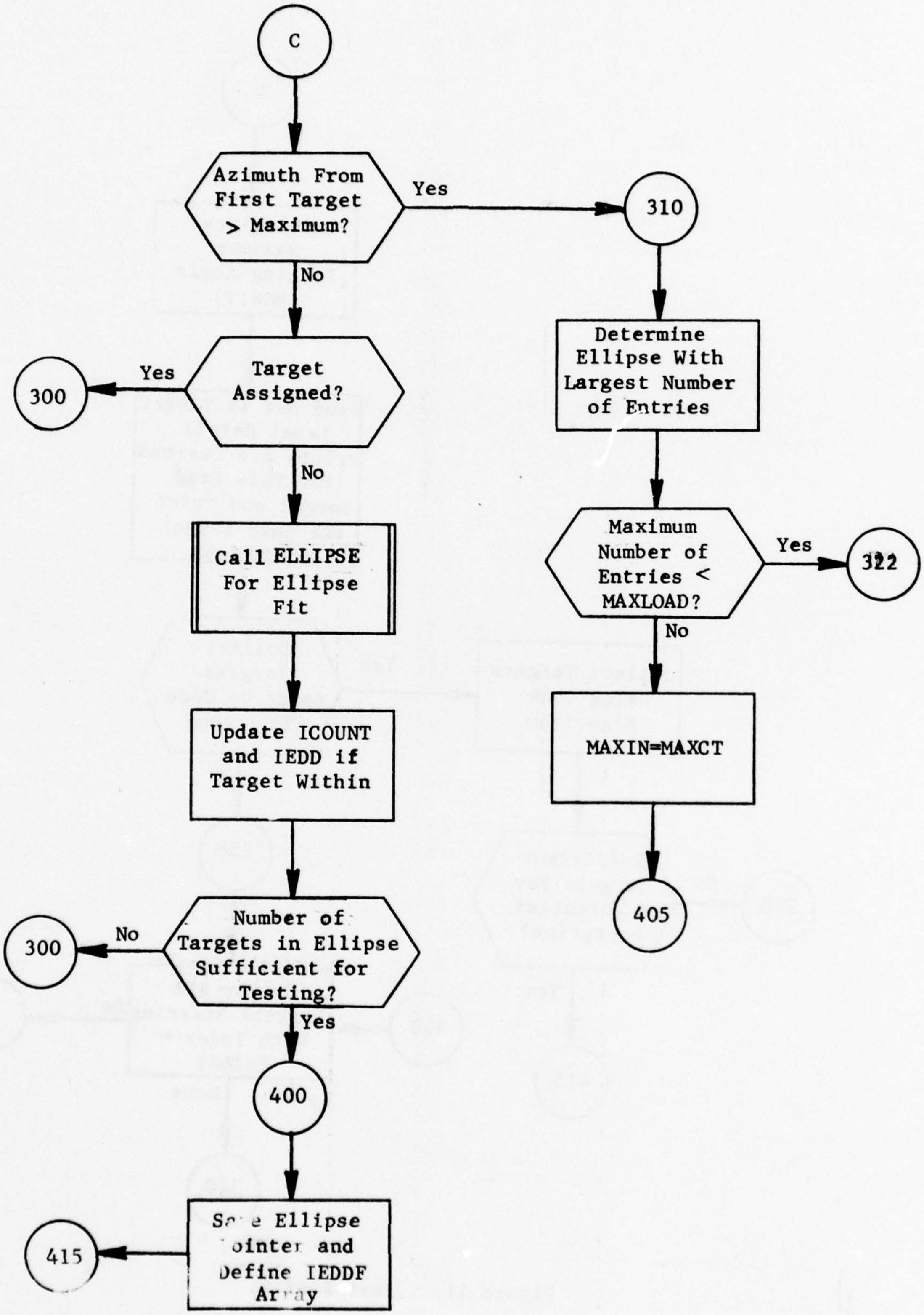


Figure 11. (Part 5 of 8)

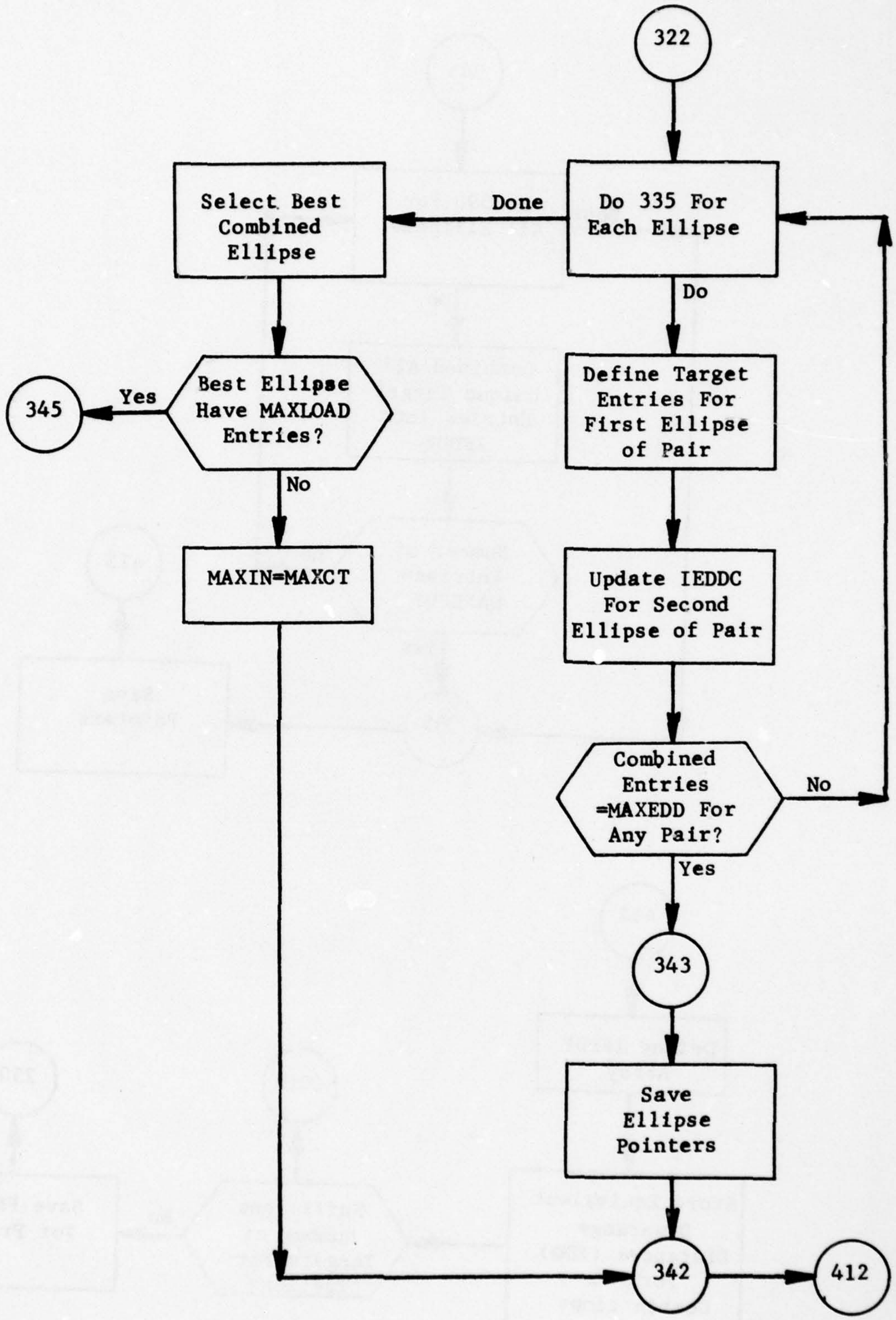


Figure 11. (Part 6 of 8)

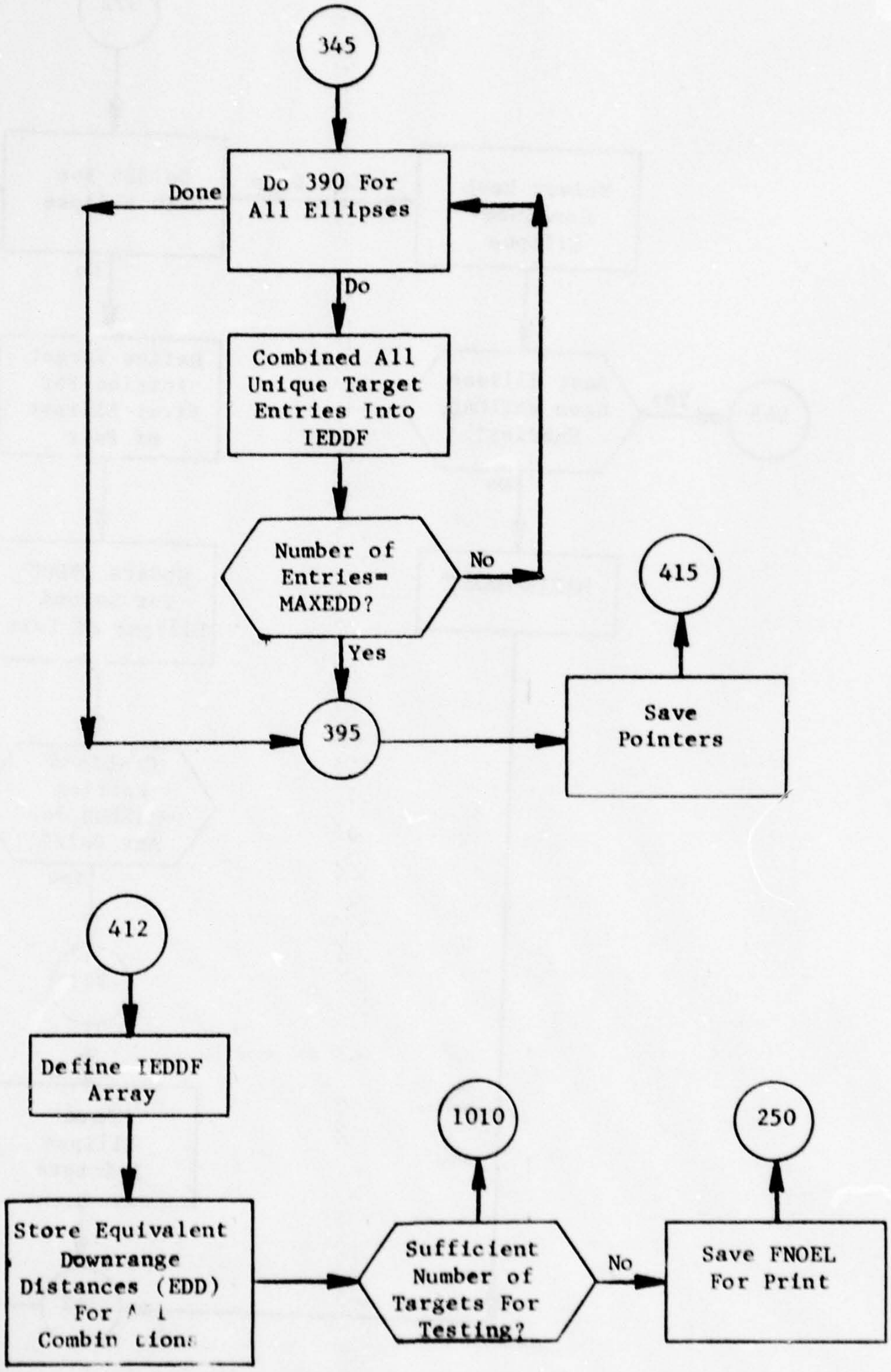


Figure 11. (Part 7 of 8)

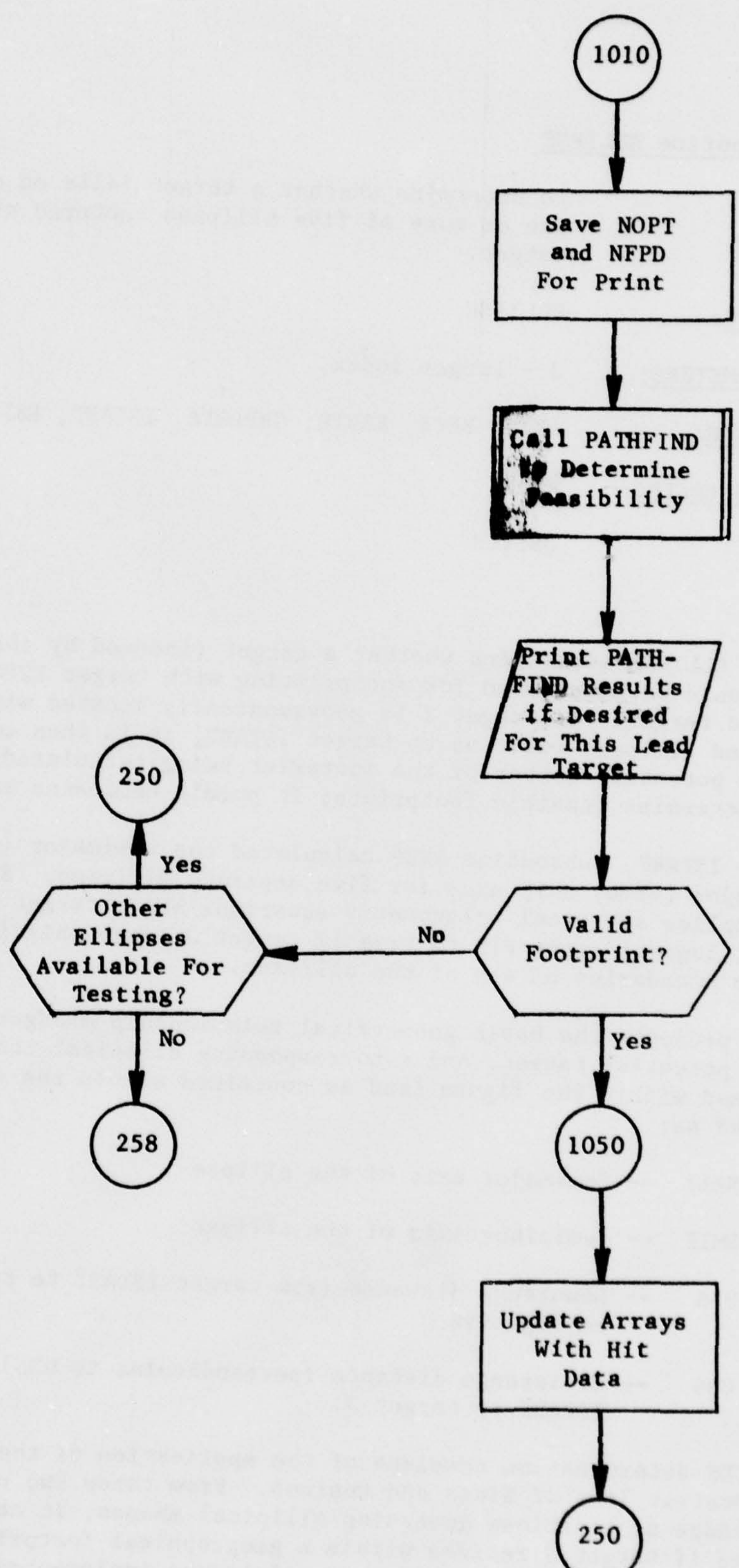


Figure 11. (Part 8 of 8)

?.6.4 Subroutine ELLIPSE

PURPOSE: To determine whether a target falls on or within one or more of five ellipses centered about a lead target.

ENTRY POINTS: ELLIPSE

FORMAL PARAMETERS: J - Target index

COMMON BLOCKS: AXIS, BETA, EARTH, GRPDATA, ISTART, RATIO, WITHIN

SUBROUTINES CALLED: None

CALLED BY: DRIVER

Method:

Subroutine ELLIPSE ascertains whether a target (indexed by formal parameter J) should be considered for footprinting with target ISTART assumed as the lead target. If target J is geographically located within one or more defined ellipses relative to target ISTART, it is then considered as being a potential member of the footprint being calculated. ELLIPSE does not determine feasible footprints; it purely nominates candidates.

For target ISTART, subroutine AXES calculated the semimajor (array SMA) and semimajor (array SMI) axes for five separate ellipses. Subroutine ELLIPSE applies spherical trignometry equations about target ISTART and J and sets logical array FIT to true if target J geographically resides within the boundaries of any of the ellipses.

Figure 12 presents the basic geometrical relationship amongst the lead target, a potential target, and a corresponding ellipical contour. Symbols as used within the figure (and as contained within the source code) are defined as:

- o SMAT -- Semimajor axis of the ellipse
- o SMIT -- Semiminor axis of the ellipse
- o UDS -- Downrange distance from target ISTART to target J. May be negative
- o CDS -- Crossrange distance (perpendicular to UDS) from target ISTART to target J.

UDS and CDS determination consists of the application of the basic spherical trigometric laws of Sines and Cosines. From these two calculations and knowledge of equations governing ellipical shapes, it can readily be determined if target J resides within a geographical footprint limit of target ISTART. Comments to follow parallels the implemented code which

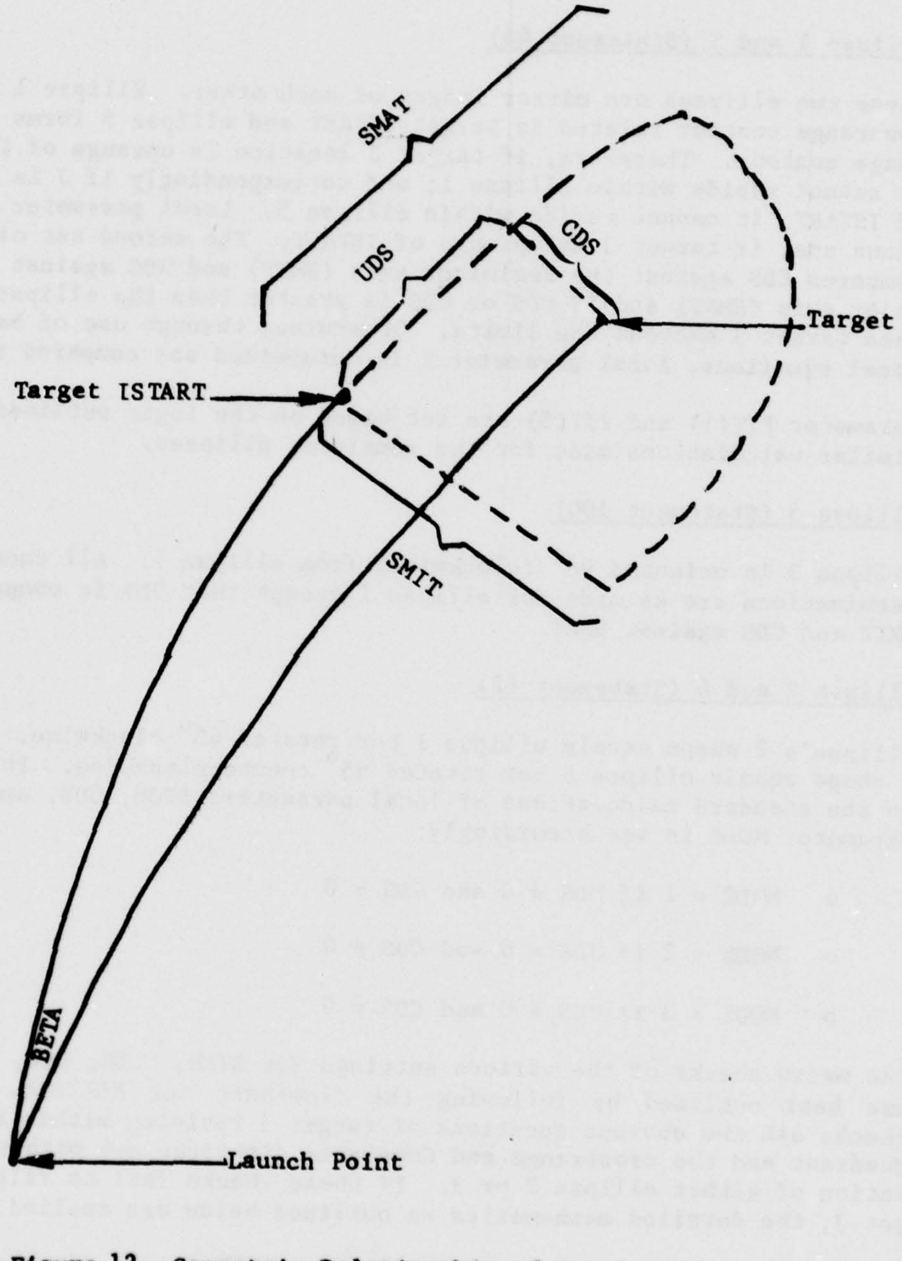


Figure 12. Geometric Relationship of Target Points Within Ellipse 1.

performs calculation on an individual ellipse basis.

Ellipse 1 and 5 (Statement 44)

These two ellipses are mirror images of each other. Ellipse 1 forms a downrange contour related to target ISTART and ellipse 5 forms an uprange contour. Therefore, if target J location is uprange of ISTART, it cannot reside within ellipse 1; and correspondingly if J is downrange of ISTART, it cannot reside within ellipse 5. Local parameter SIGM equals minus one, if target J is uprange of ISTART. The second set of checks compares CDS against the semiminor axis (SMIT) and UDS against the semimajor axis (SMAT) and if CDS or UDS is greater than the ellipse parameter then target J exceeds the limits. Otherwise, through use of basic elliptical equations, local parameter X is determined and compared accordingly.

Parameter FIT(1) and FIT(5) are set based on the logic outlined above and similar calculations made for the remaining ellipses.

Ellipse 3 (Statement 100)

Ellipse 3 is oriented 90° (clockwise) from ellipse 1. All checks and determinations are as made for ellipse 1 except that UDS is compared against SMIT and CDS against SMAT.

Ellipse 2 and 4 (Statement 62)

Ellipse's 2 shape equals ellipse 1 but rotated 45° clockwise. Ellipse's 4 shape equals ellipse 5 but rotated 45° counterclockwise. In addition to the standard calculations of local parameters SIGM, UDS, and CDS, parameter MODE is set accordingly:

- o MODE = 1 if UDS $\neq$ 0 and CDS = 0
- o MODE = 2 if UDS = 0 and CDS $\neq$ 0
- o MODE = 3 if UDS $\neq$ 0 and CDS $\neq$ 0

The macro checks of the various settings for SIGM, MODE, UDS, and CDS are best outlined by following the flowchart for ELLIPSE. These checks ask the obvious questions of target J residing within the correct quadrant and the crossrange and downrange distances are within the limitation of either ellipse 2 or 3. If these checks fail to filter out target J, the detailed mathematics as outlined below are applied.

Figure 13 presents the pictural relationship of target spacing relative the the ellipse 2 coordinate system. The dash curve represents ellipse's 2 contour where target J must reside for footprinting consideration. Parameter C is determined from the law of Cosines; parameter ALF from the law of Sines. Knowing ALF and knowing ellipse 2 is rotated 45°, parameter ALT is readily determined and APY and BPX now calculated from the laws of Sines and Cosines respectively.

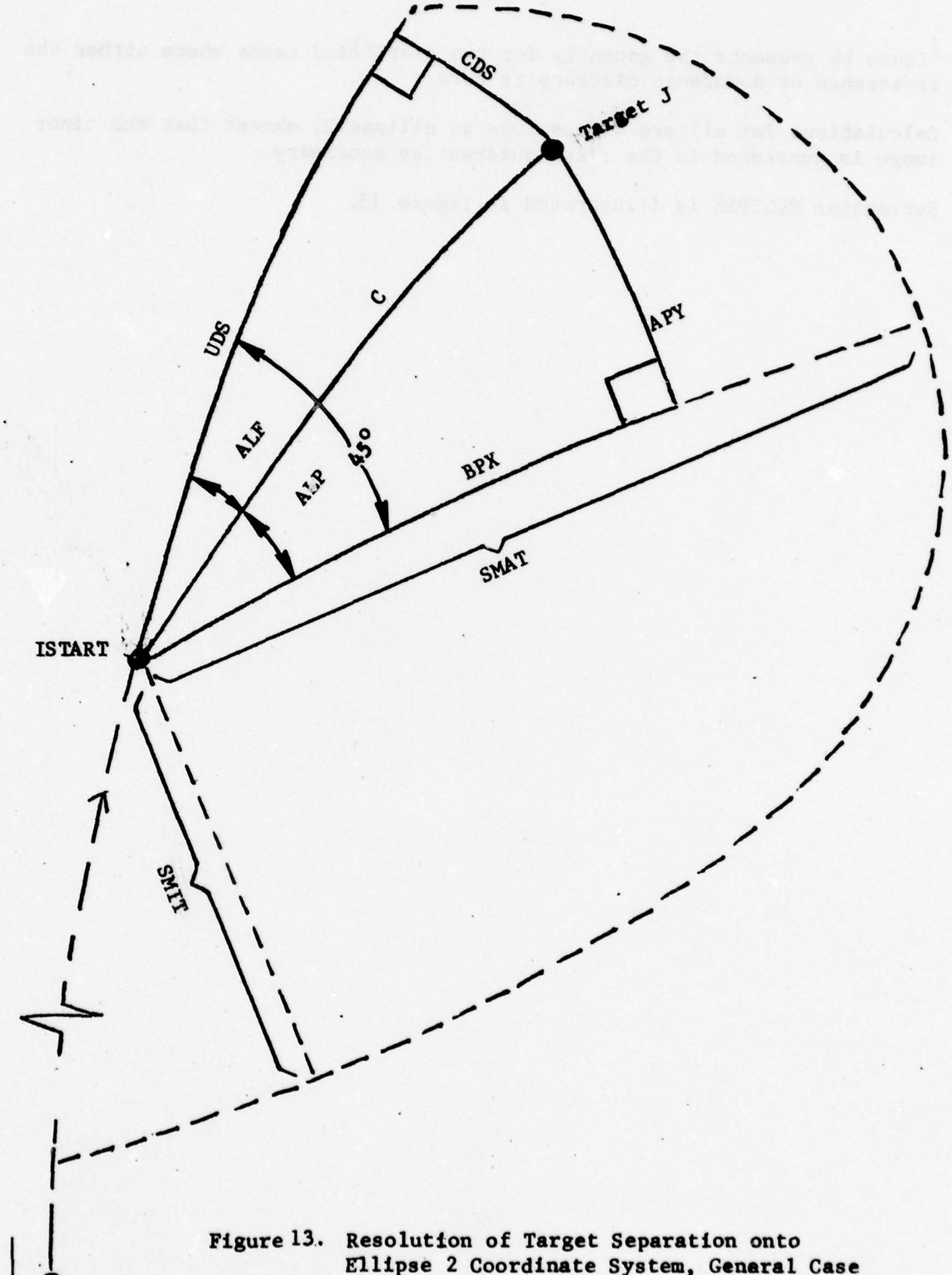
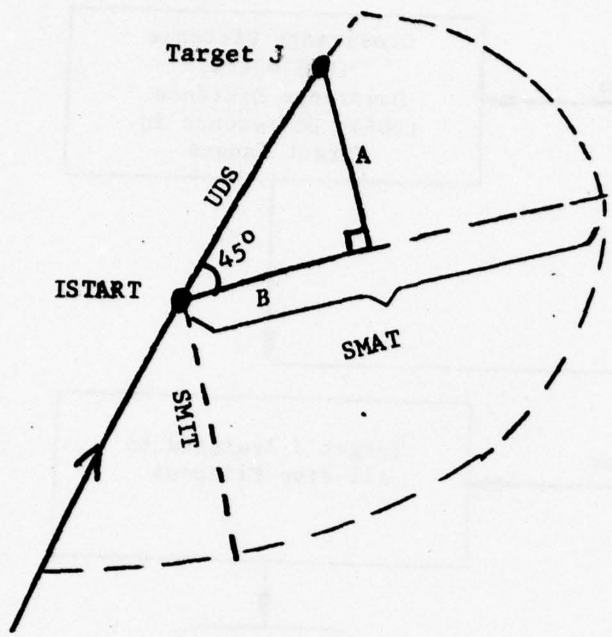


Figure 13. Resolution of Target Separation onto Ellipse 2 Coordinate System, General Case

Figure 14 presents the geometry for two simplified cases where either the crossrange or downrange distance is zero.

Calculations for ellipse 4 is solved as ellipse 2, except that the minor image is generated in the first quadrant as necessary.

Subroutine ELLIPSE is illustrated in figure 15.



MODE 1:

When Crossrange Distance is Zero, Downrange Distance, UDS, becomes Triangle Side C

MODE 2:

When Downrange Distance is Zero, Crossrange Distance, CDS, becomes Triangle Side C

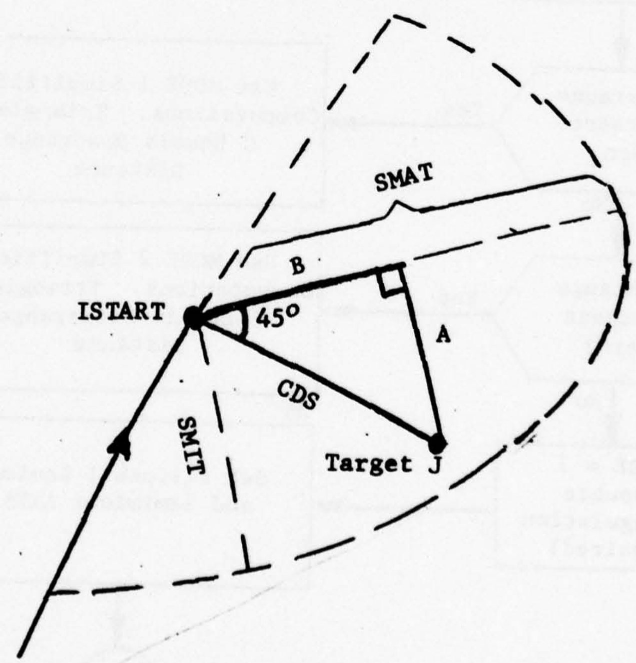


Figure 14. Resolution of Target Separation onto Ellipse 2 Coordinate System, Simplified Cases

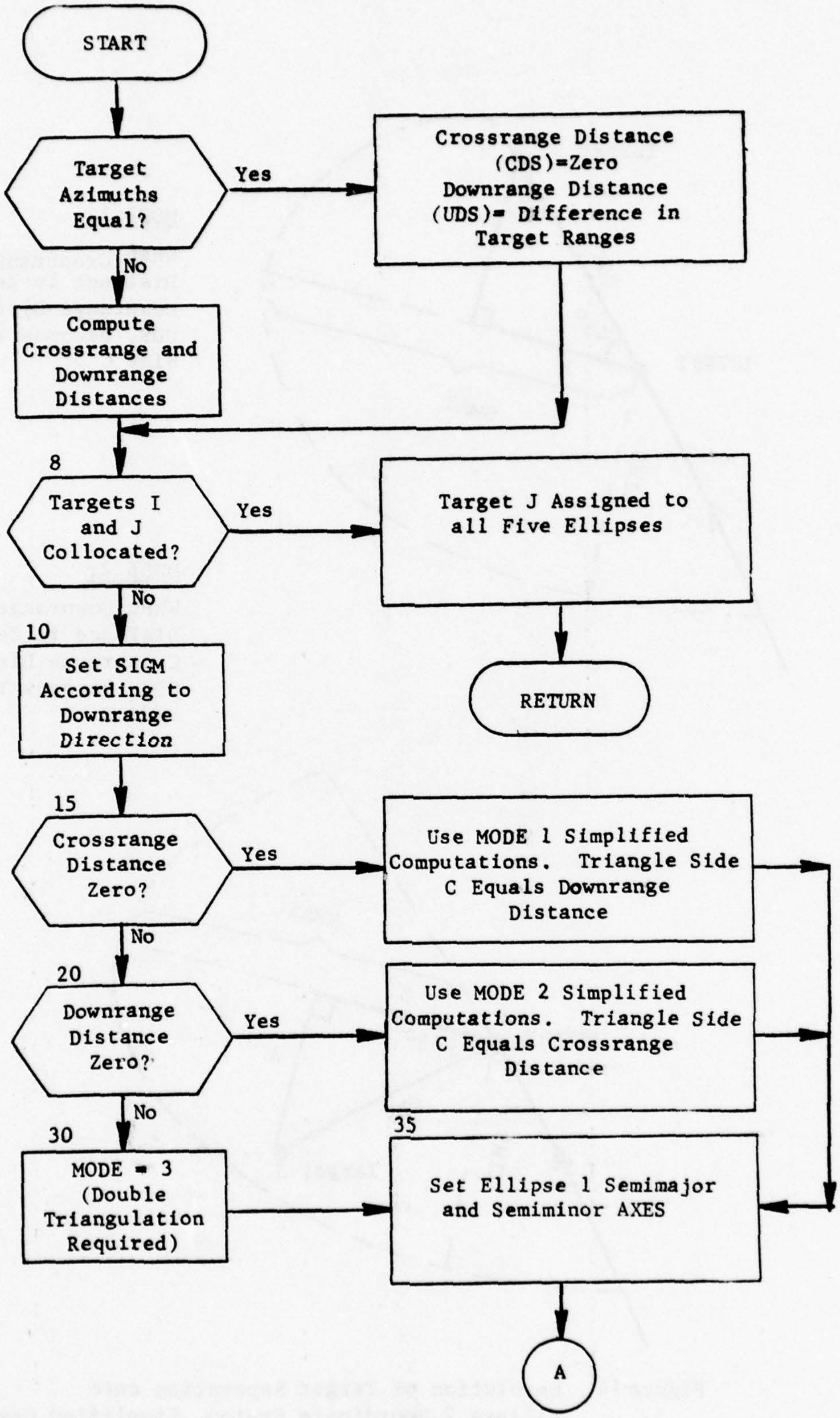


Figure 15. Subroutine ELLIPSE (Part 1 of 10)

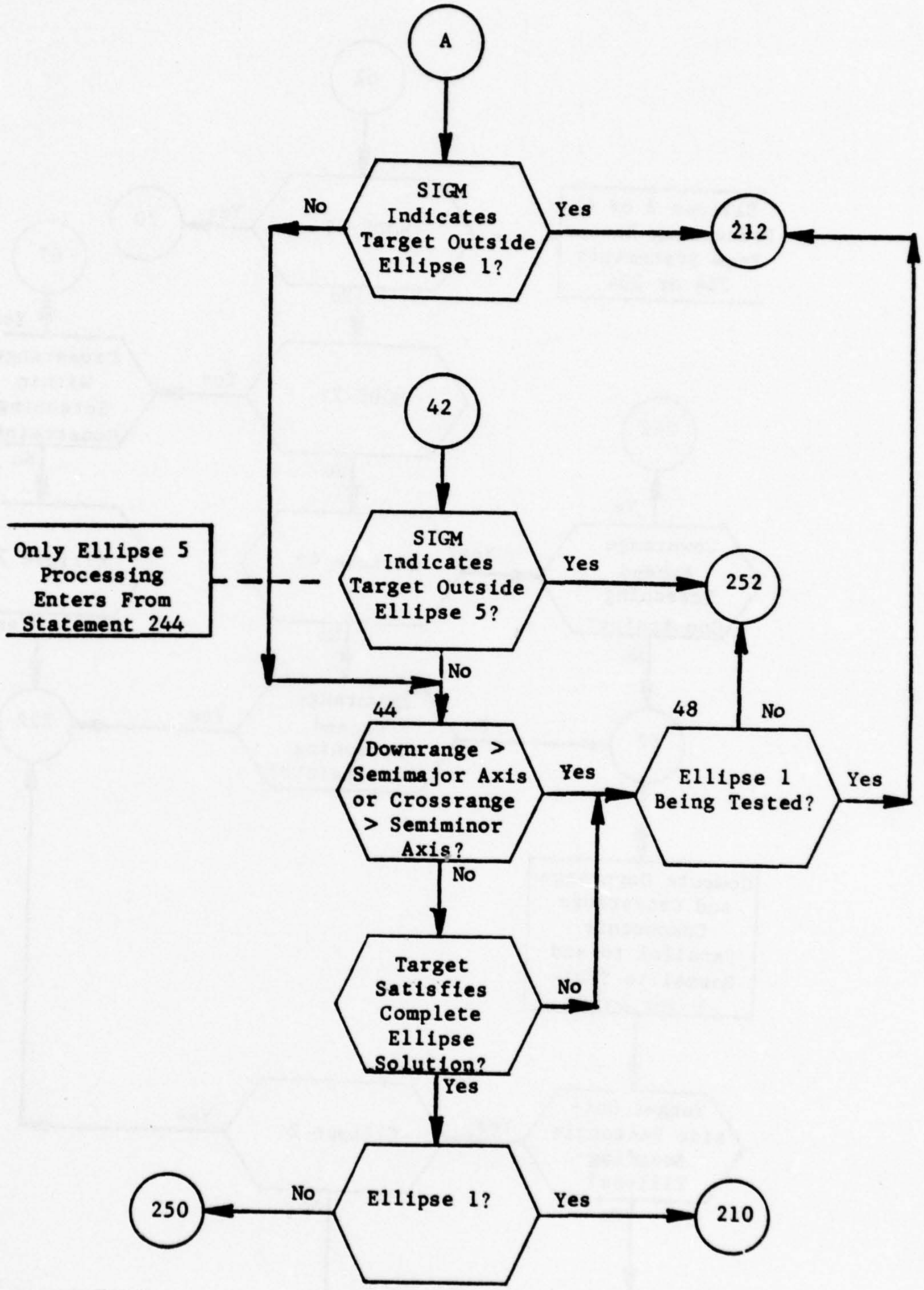


Figure 15. (Part 2 of 10)

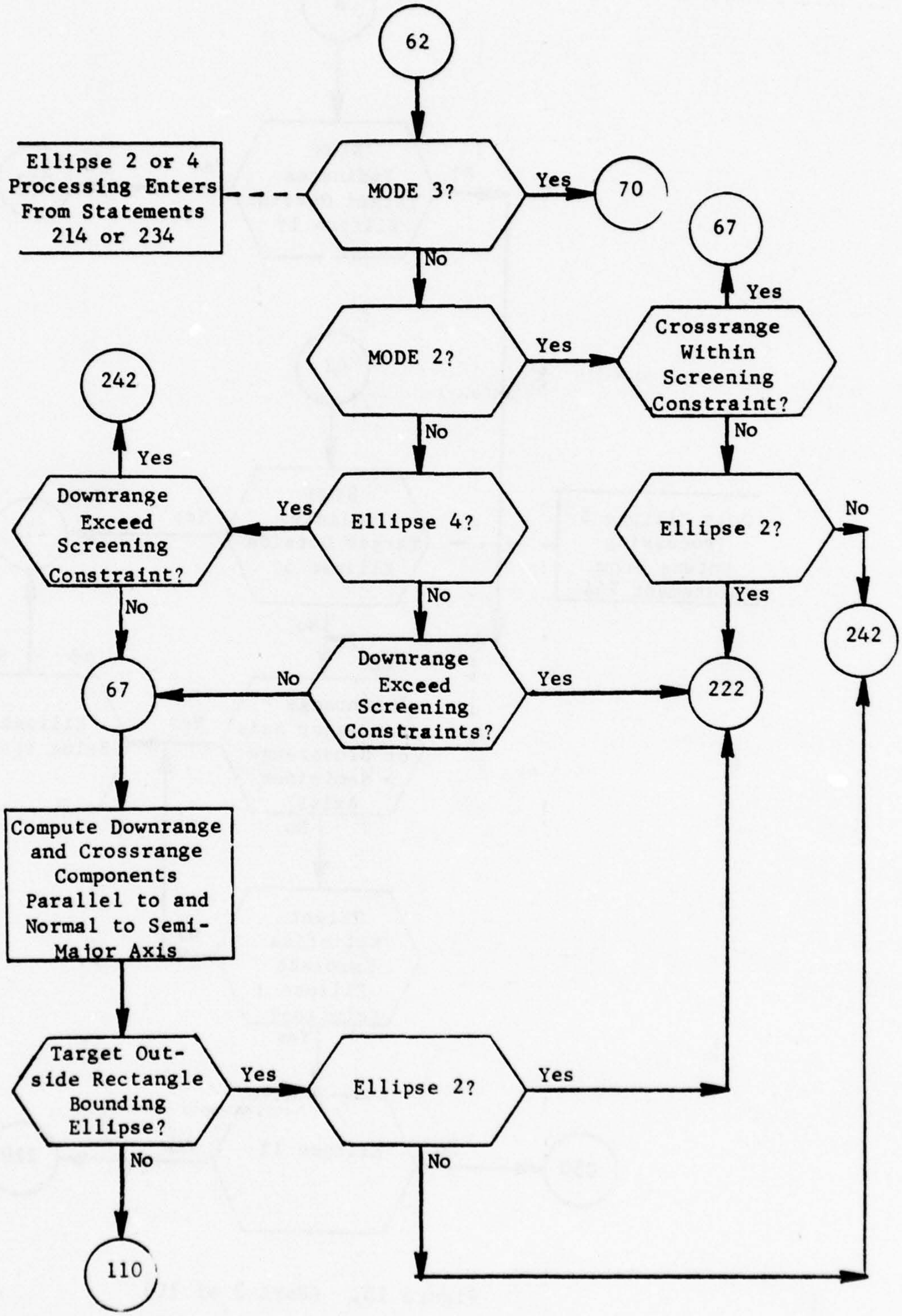


Figure 15. (Part 3 of 10)

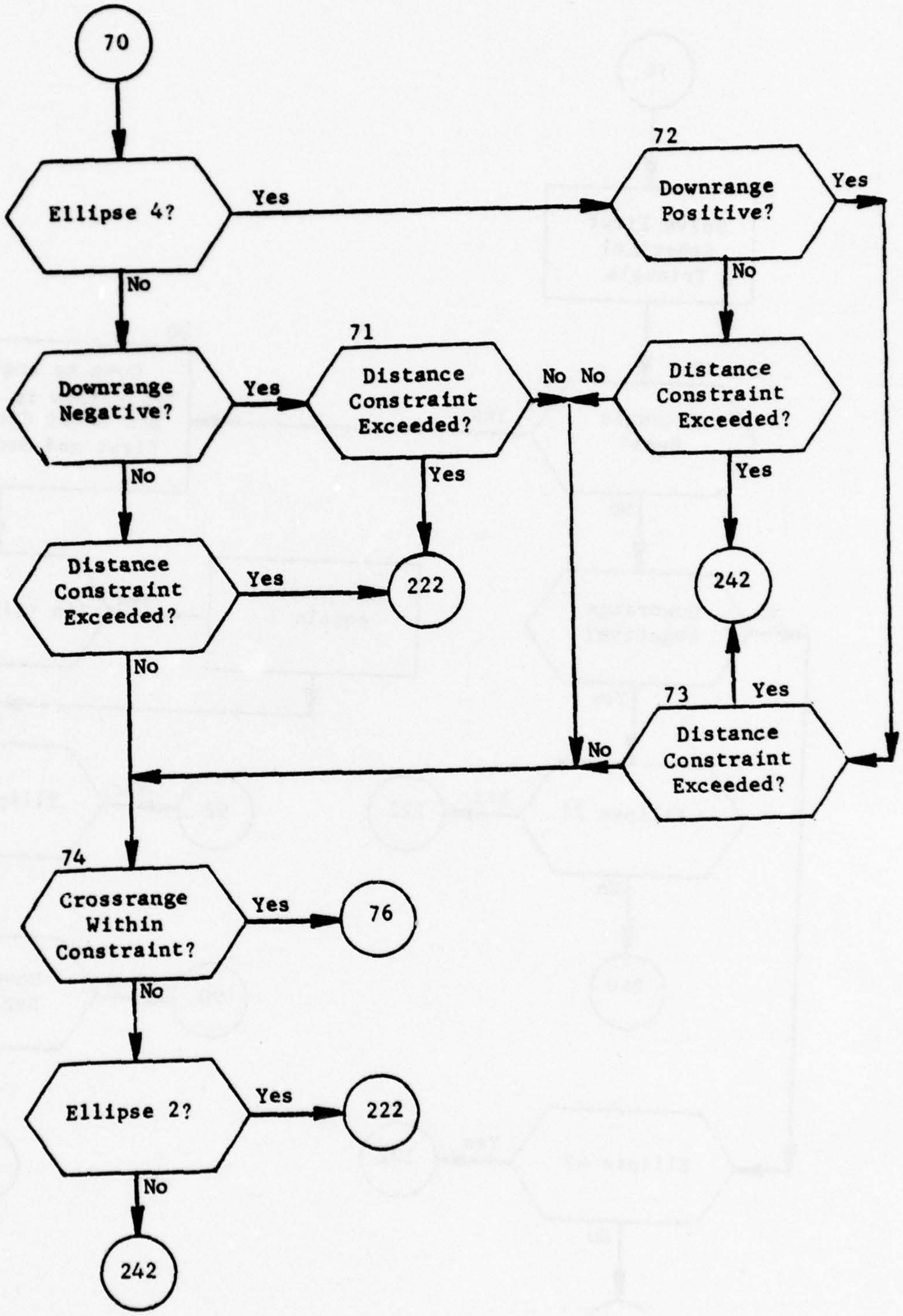


Figure 15. (Part 4 of 10)

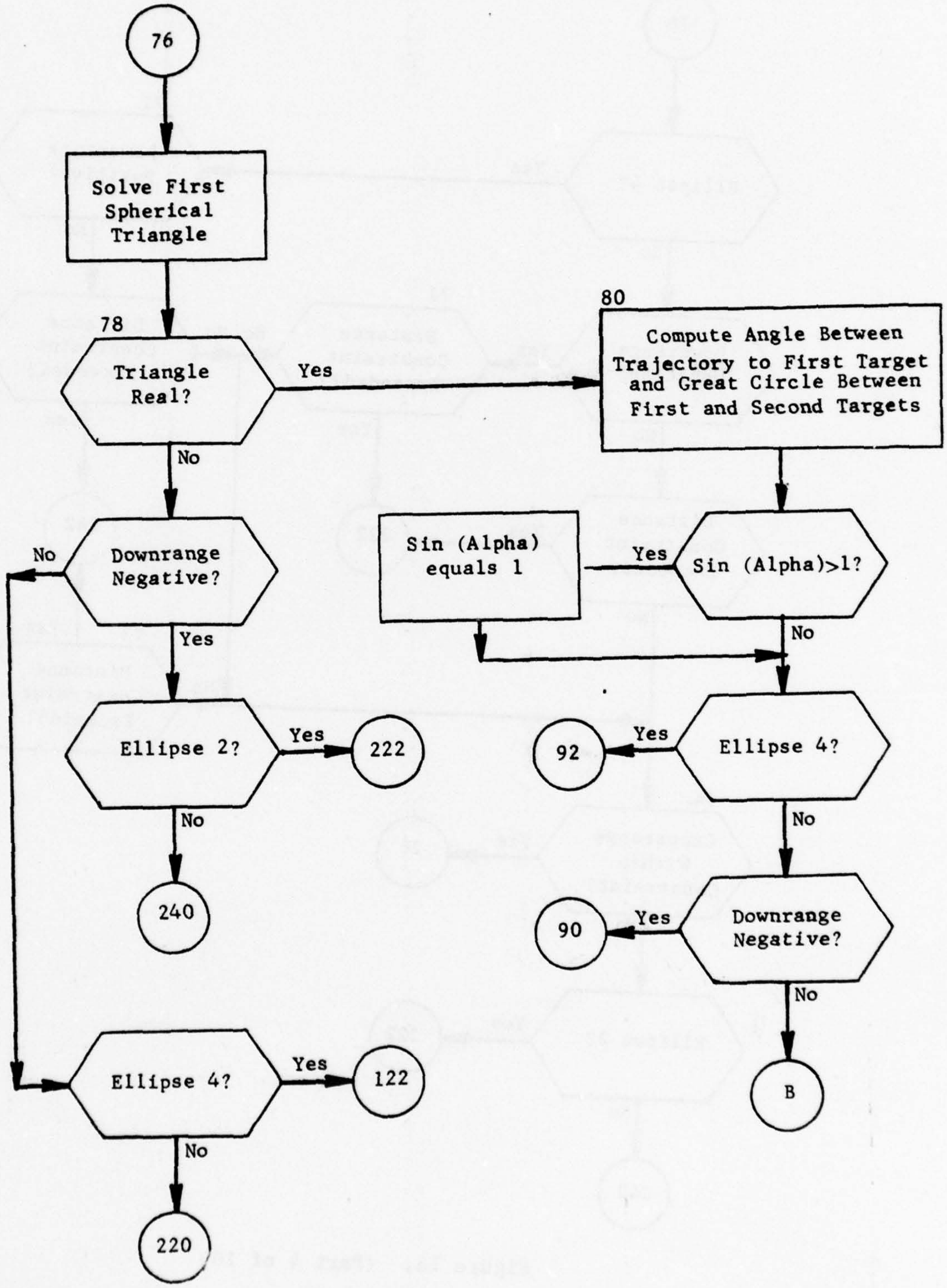


Figure 15. (Part 5 of 10)

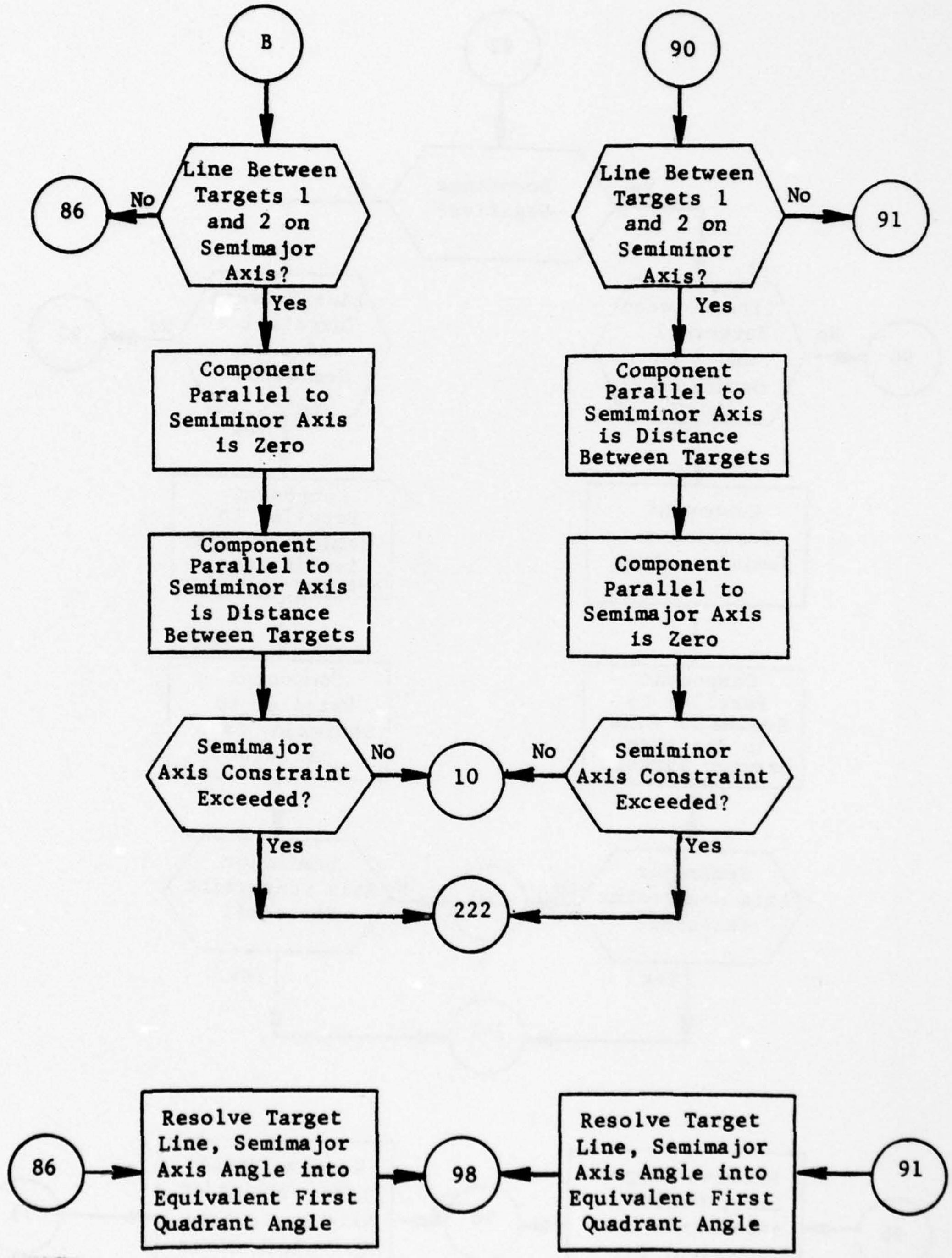


Figure 15. (Part 6 of 10)

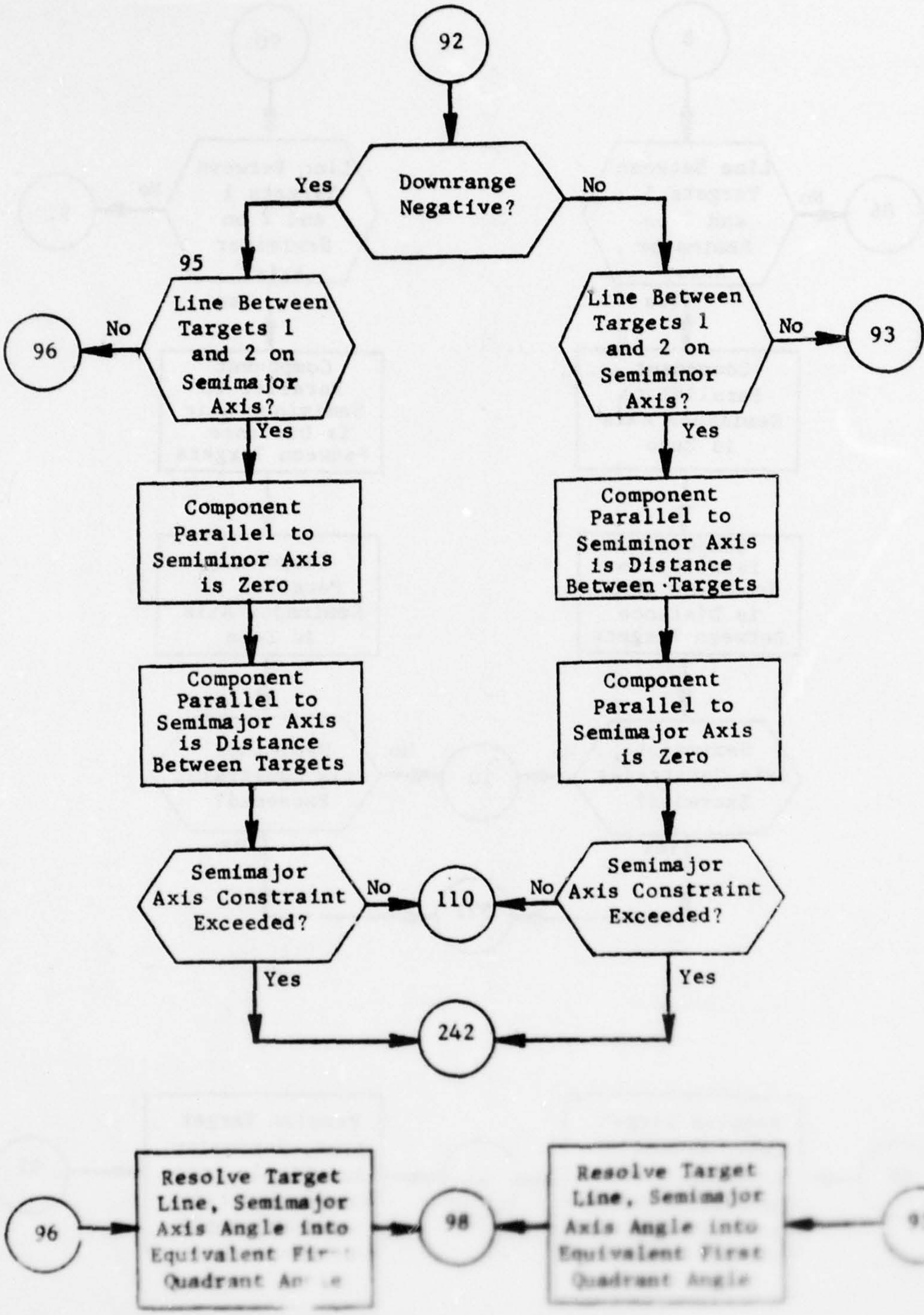


Figure 15. (Part 7 of 10)

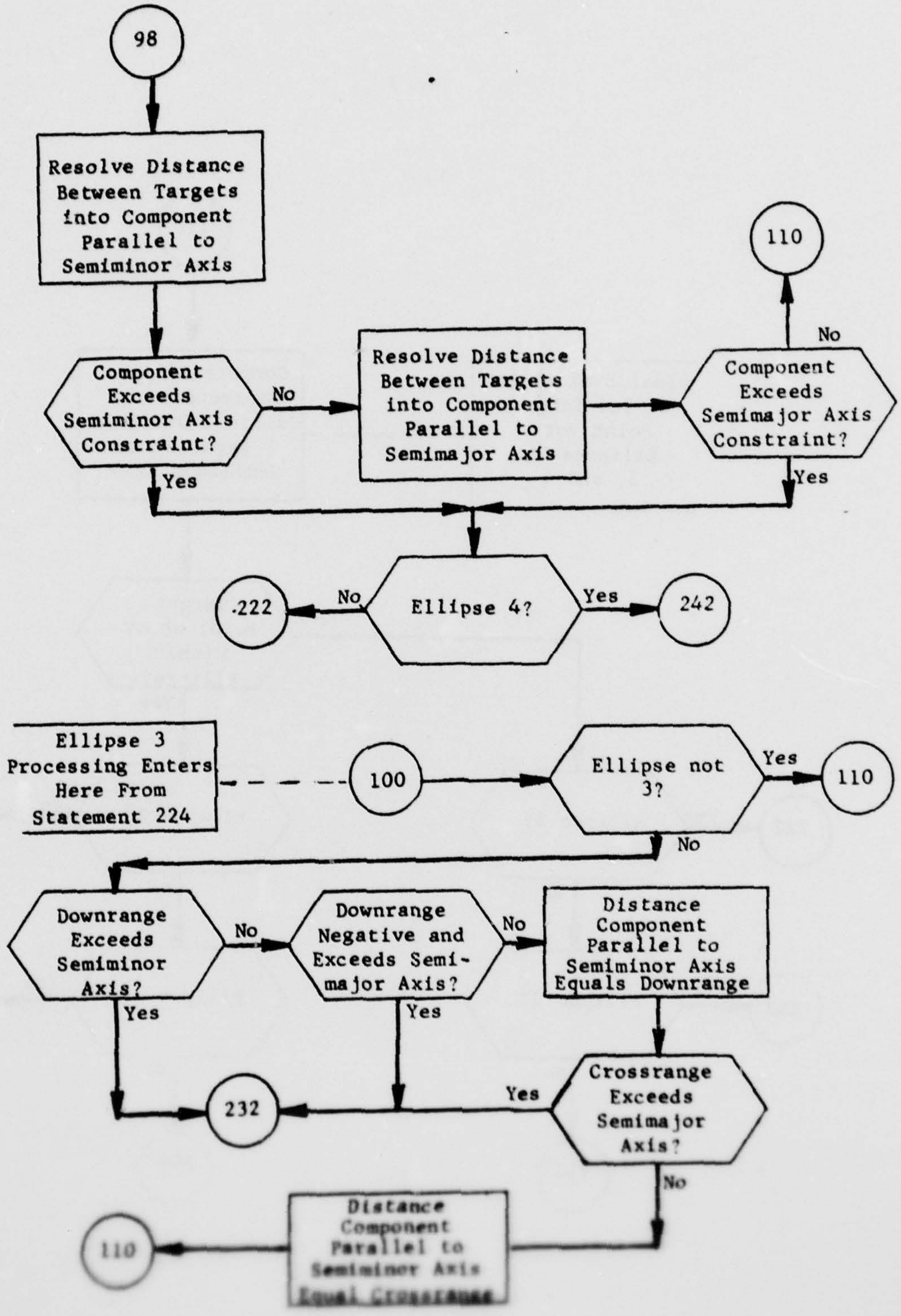
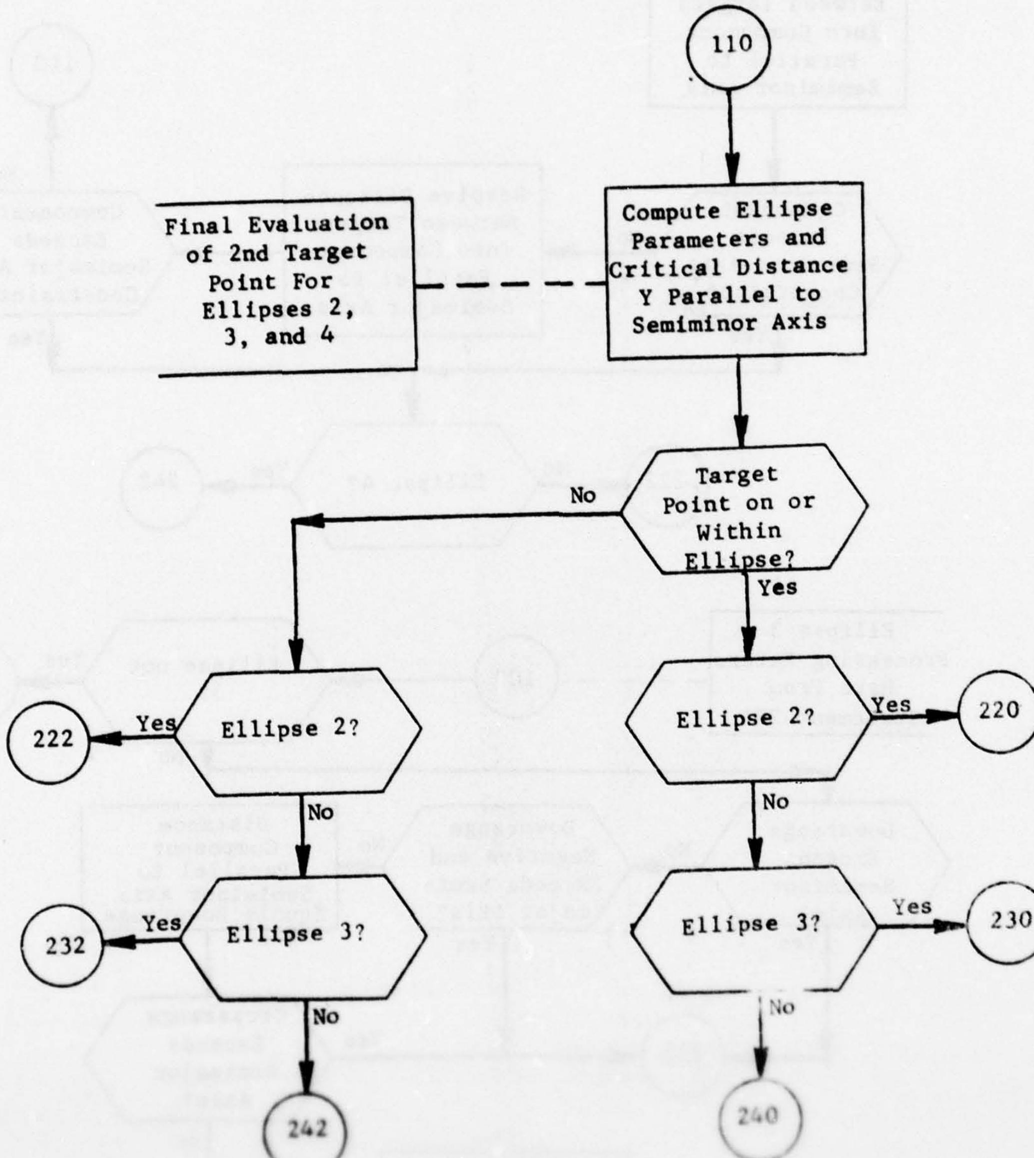


Figure 13. (Part 8 of 10)


110
Compute Ellipse Parameters and Critical Distance Y Parallel to Semiminor Axis
Final Evaluation of 2nd Target Point For Ellipses 2, 3, and 4
Target Point on or Within Ellipse?
No
Yes
Ellipse 2?
Yes
No
222
220
Ellipse 3?
Yes
No
232
230
242
240

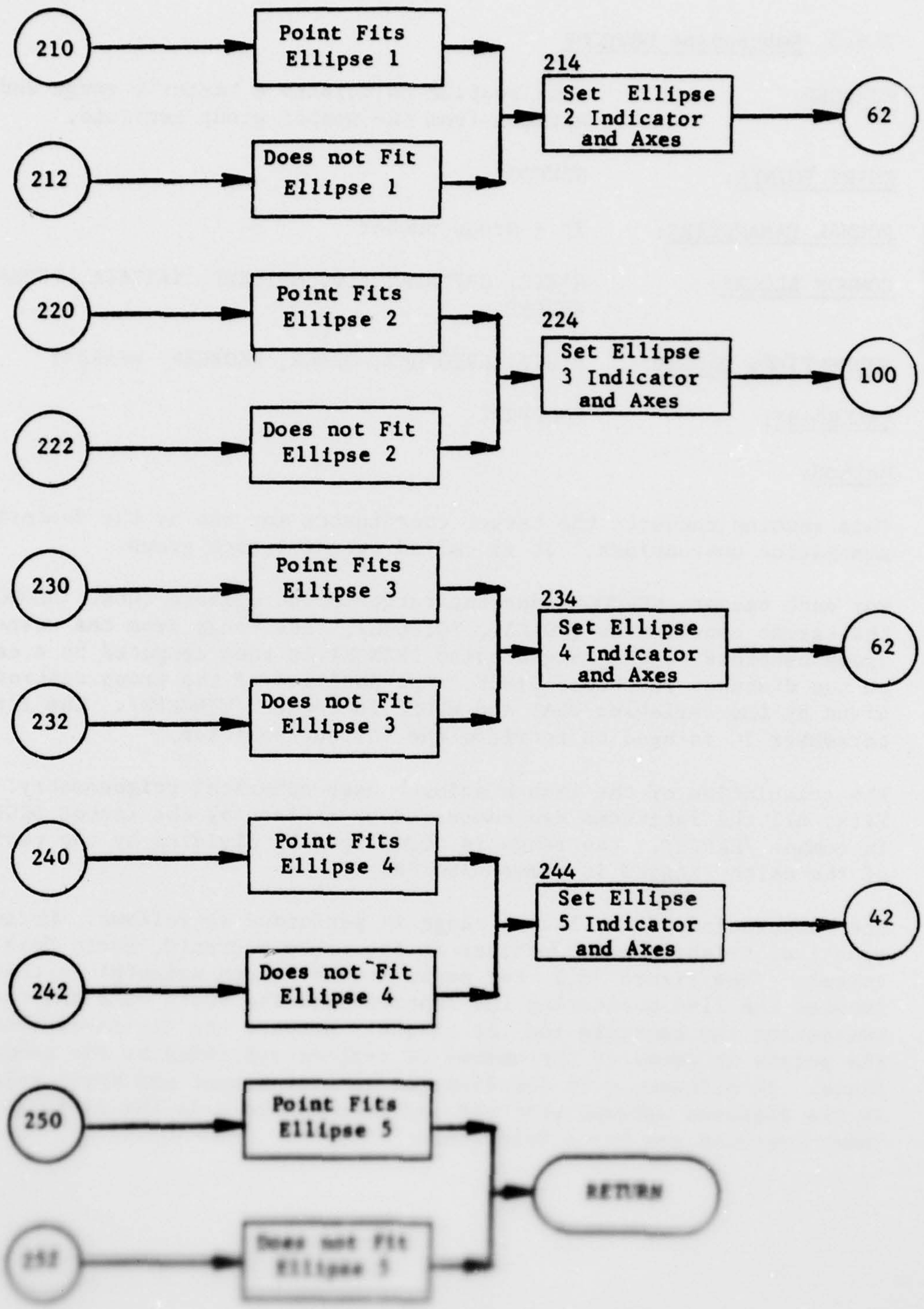


Figure 11. (Part 10 of 10)

2.6.5 Subroutine NEWCOOR

PURPOSE: This routine calculates a target's range and azimuth from the weapon group centroid.

ENTRY POINTS: NEWCOOR

FORMAL PARAMETERS: IG - Group number

COMMON BLOCKS: EARTH, GRPDATA, ITP, MYIDENT, RAIDATA, TSCRATCH, WPNGRP

SUBROUTINES CALLED: DISTF, LREORDER, ORDER, REORDER, WRARRAY

CALLED BY: FOOTPRNT

Method:

This routine converts the target coordinates for use by the footprint generation subroutines. It is called once for each group.

For each target, NEWCOOR adds the target point offsets (DLAT, DLONG) to the target coordinates (TGTLAT, TGTLONG). The range from the weapon group centroid to the target point (RANGE) is then computed by a call on the distance function, DISTF. The position of the group centroid is given by the variables WLAT and WLONG in common /WPNGRPX/. The formal parameter IG is used to retrieve the correct position.

The calculation of the launch azimuth uses spherical trigonometry. First all the latitudes are converted to radians by the factor DEGTORAD in common /EARTH/. The range is normalized by dividing by the radius of the earth (RADIUS in common /EARTH/).

The computation of the launch range is performed as follows. Define a spherical triangle with vertices at the group centroid, North Pole and target. (See figure 16.) Let angle A (the launch azimuth) be the angle between the line connecting the centroid and the North Pole and the line connecting the centroid and the target. Measure the distances between the points in terms of the number of radians subtended by the connecting lines. If distance a is the distance between target and North Pole, b is the distance between centroid and target, and c is the distance between centroid and North Pole, then:

$a = \pi/2 - (\text{TGTLAT*DEGTORAD})$

$b = \text{RANGE/RADIUS}$

$c = \pi/2 - (\text{WLAT*DEGTORAD})$

Using the law of cosines for spherical triangles, then:

$$\cos A = \frac{\cos a - (\cos b \cdot \cos c)}{\sin b \cdot \sin c}$$

The difference between the target longitude and the centroid longitude is then used to determine the sign of the launch azimuth.

After all launch azimuths have been computed, the target data are reordered according to increasing value of launch azimuth. The sequence array used for this reordering is written on the assignment data scratch file, ISCR, for later use.

Subroutine NEWCOOR is illustrated in figure 17.

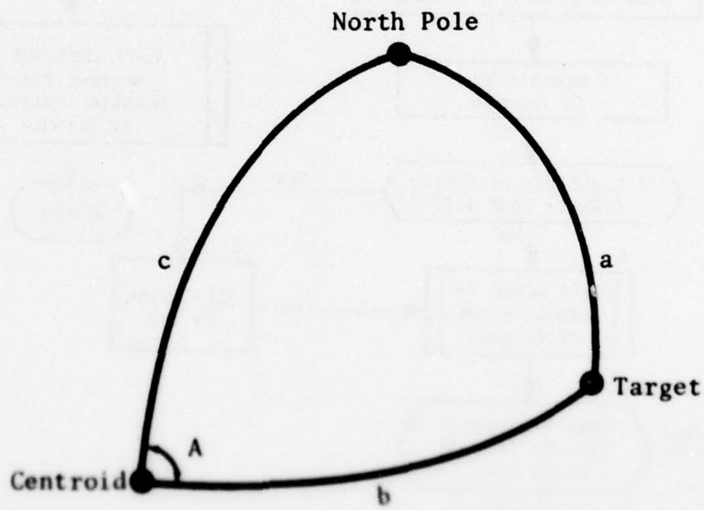


Figure 16. Calculation of Launch Azimuth

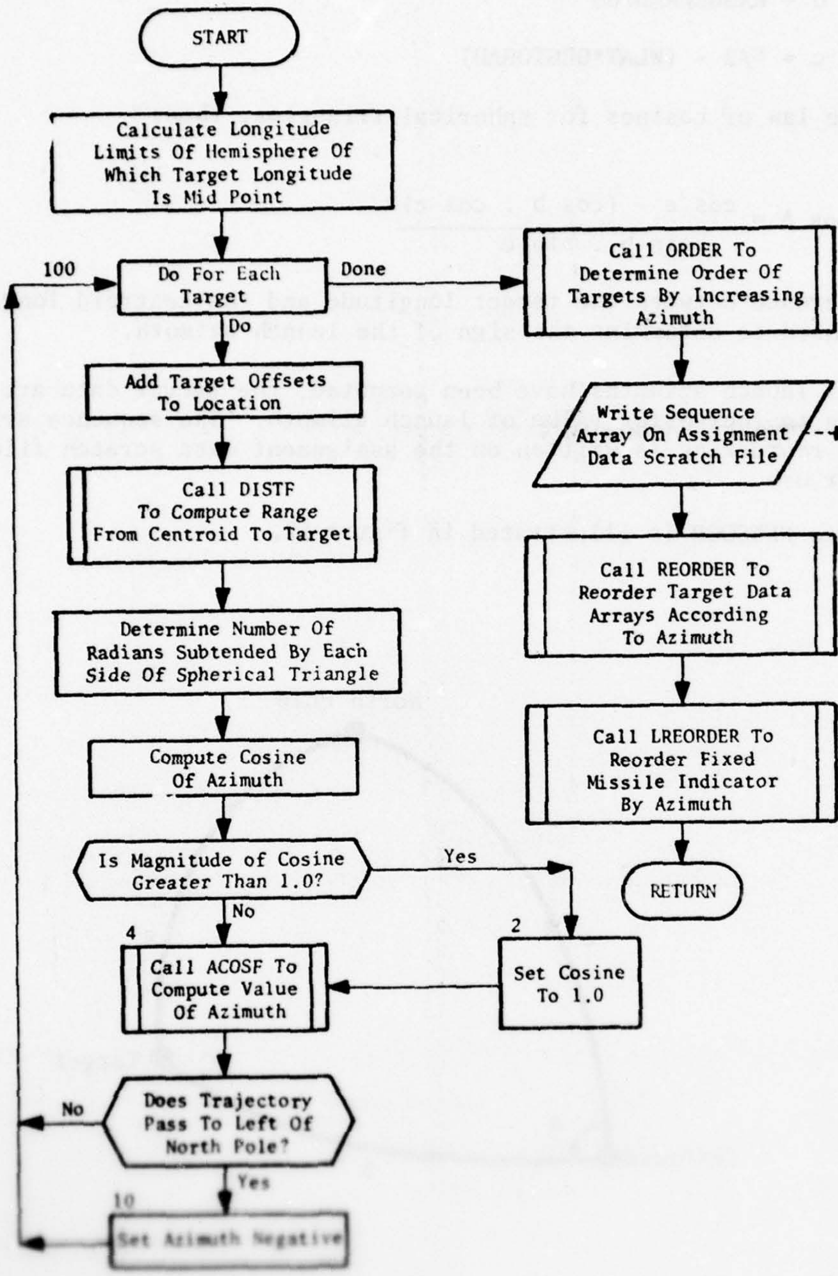


Figure 17. Subroutine

2.6.6 Subroutine PATHFIND

PURPOSE:	This routine will attempt to find a feasible footprint from a collection of potential targets.
ENTRY POINTS:	PATHFIND
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ALGOS, CSAVE, EDD, HIT, ISTART, MAXPB, PATHS RAIDATA, STATS, SYSMAX
SUBROUTINES CALLED:	GLOG, SLOG
CALLED BY:	DRIVER

Method:

Subroutine DRIVER executes PATHFIND upon collecting a subset of targets that may potentially form a feasible footprint. PATHFIND interrogates the target list and finds an acceptable "path", if it exists, which when traversed will hit the proper number of targets within the fuel constraints of the MIRV platform; hence forming a footprint.

Subroutine DRIVER supplies:

- EDD(I,J) -- Effective distance between all target I,J combinations.
- FACT(I) -- The MIRV fuel consumption rate based on the number of RVs on board (referred to as stages). FACT(1) is the rate with one RV on board.
- MAXLOAD -- The loading factor for the MIRV platforms.
- MAXIN -- Number of entries supplied by DRIVER. Cannot be less than MAXLOAD.
- FUELOAD -- Amount of fuel initially on board.

PATHFIND conducts operations in three phases: initialization and predynamic algorithms; dynamic programming algorithm and branch and bound algorithms. Elaboration on each phase follows.

Initialization and Predynamic Algorithm Phase

This phase prepares parameters for the other phases and performs simplified checks for obvious paths that may yield a feasible footprint.

In a sequential manner the following are initialized.

- o IND -- Number of targets considered
- o ISTG -- Number of stages (or number of RVs onboard)
- o ISTG1 -- Number of stages required to deliver all RVs
- o NTOUR -- Number of optimum solutions found
- o SUBS(I,J) -- Target index of subtour for pass, branch combinations (subtour, pass, branch explained under the branch and bound algorithms)
- o SUBL(I,J) -- Number of legs in subtour
- o COST(I,J) -- Cost of subtour
- o NOLEG(I,J) -- Indicator of no legs within a subtour
- o IPATH(I,J,K) -- Target visited each stage of subtour
- o LXNBF(K) -- Indicator that target was never branched from
- o LXBNP(K) -- Indicator of target to be branched from on the next pass
- o LXNOSUB(K) -- Indicator of no subtour on this branch
- o LXOPT(K) -- Indicator that branch is optimal solution
- o LXMCOST(K) -- Indicator that branch has minimum cost
- o IX(I,J) -- Points to next target to be visited for stage I and current target J
- o TGT(I) -- Target index number
- o HIT(I) -- Target index numbers of the final path
- o MORE -- Indicator if more branches exist
- o IBNEW -- Number of branches for next pass
- o II -- Pass number
- o IB -- Pass being processed to find branches
- o JJ -- Branch number
- o IB -- Branch being processed for further branching
- o BNO -- Number of branches in subtour counter
- o NHIT -- Target hit counter
- o X(I,J) -- Working array for effective inter target distance. If any target collocated, set to a large value (parameter BIL) to prevent assigning twice within a footprint
- o IPASS -- Pass counter
- o IBMAX -- Branch counter for this pass
- o [illegible] -- [illegible]

Immediately following the initializing of required parameters two simplified observations of the target list (DO loops 188 and 5050) attempt to develop a feasible footprint.

DO loop 188 collects costs assuming the path sequence consists of starting with target one, going to target two, continuing to target three and continuing the numerical sequence up to ISTG1 stages. If this collection costs less than FUELOAD, it is called a feasible footprint. Note within the DO loop of the use of indexes for arrays X and FACT. A sequence of 1-2-3----ISTG requires fuel consumption rates for ISTG, ISTG-1, ISTG-2, ---1 RVs onboard which explains the reversal of indexes.

If DO loop 188 final cost exceeded FUELOAD, a second degree of sophistication is applied within DO loop 5050. This loop assumes target one as being the first target within the path and from that assumption each remaining potential target is queried to find the minimal cost of going from target one to the second stage. Minimal costs of going from stage N to stage N+1 are now found. For each consideration, a path that repeats a previously assigned target is denied. For each stage all remaining targets are considered in obtaining the minimal costs (parameter AID) of going to the next stage. As always, costs are accumulated (parameter XCOST) and upon completion compared against FUELOAD.

These two simplified searches reduces computer time by finding the obvious solutions. Note that these solutions adhere to the objective of finding feasible solutions but not necessarily optimal solutions (in terms of finding a path that utilizes minimal fuel costs). Optimal paths are necessary only if no other path produces a solution.

<u>Dynamic Programming Algorithm</u>

The dynamic programming algorithm begins with label 48 (refer to figure 18) and all logic contained within DO loop 1000.

This solution is a variation of the classical traveling salesman which requires a salesman to visit N cities once and only once in a path such that he visits the cities with minimal total distance traveled. For the footprint problem cities become target assignments and distance becomes the fuel costs of going from assignment I to J.

The solution consists of working the problem backward. Starting at the end condition (one RV onboard) the minimal path of arriving there is obtained by querying the costs of all potential second stage targets. Once found, the second stage (two RVs onboard) is processed to find the optimal I,J third stage target combination. This method continues until each I,J minimal combination is obtained for all onboard stages. For each stage, arrays ST and IX are stored. ST(I,J) contains the minimal accumulated cost for stage I and target J. Array IX(I,J) stores the best next target (K) for stage I and target J. Therefore, for each stage the accumulated cost and best path is contained within ST and IX. The global optimal, then, is a summation of local optimals of going from one stage to the next.

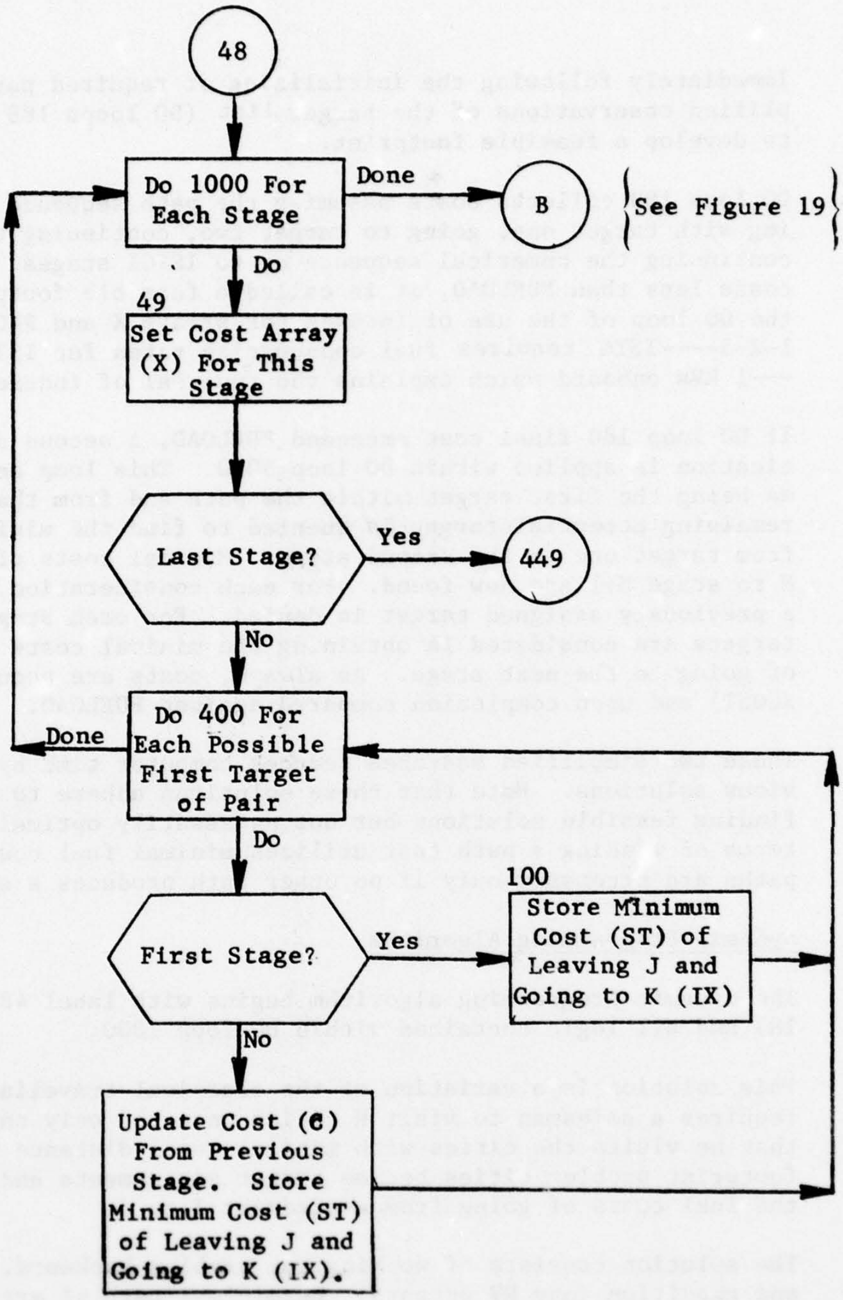


Figure 18. Dynamic Programming Algorithm (Part 1 of 2)

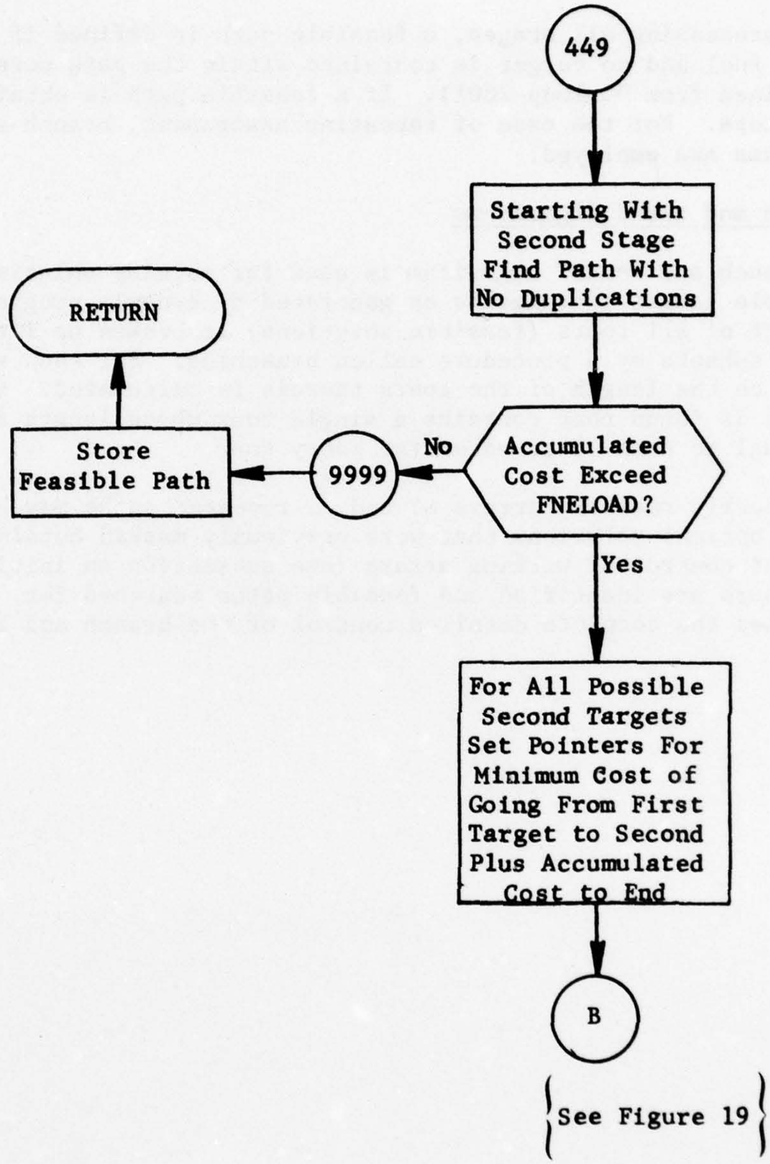


Figure 18. (Part 2 of 2)

Upon processing all stages, a feasible path is defined if there is sufficient fuel and no target is contained within the path more than once (obtained from DO loop 2001). If a feasible path is obtained, processing stops. For the case of repeating assignment, branch and bound algorithms are employed.

Branch and Bound Algorithms

A "branch and bound" algorithm is used for solving solutions containing multiple target assignments as generated by dynamic programming exists. The set of all tours (feasible solutions) is broken up into increasingly small subsets by a procedure called branching. For each subset a lower bound on the length of the tours therein is calculated. Eventually, a subset is found that contains a single tour whose length is less than or equal to some lower bound for every tour.

By properly resettig arrays ST and X, repeated paths may be denied and other optimal solutions that were previously masked obtained. Through correct control of working arrays (see subsection on initialization) new tours are identified and feasible paths searched for. Figure 19 outlines the complete detailed control of the branch and bound algorithms.

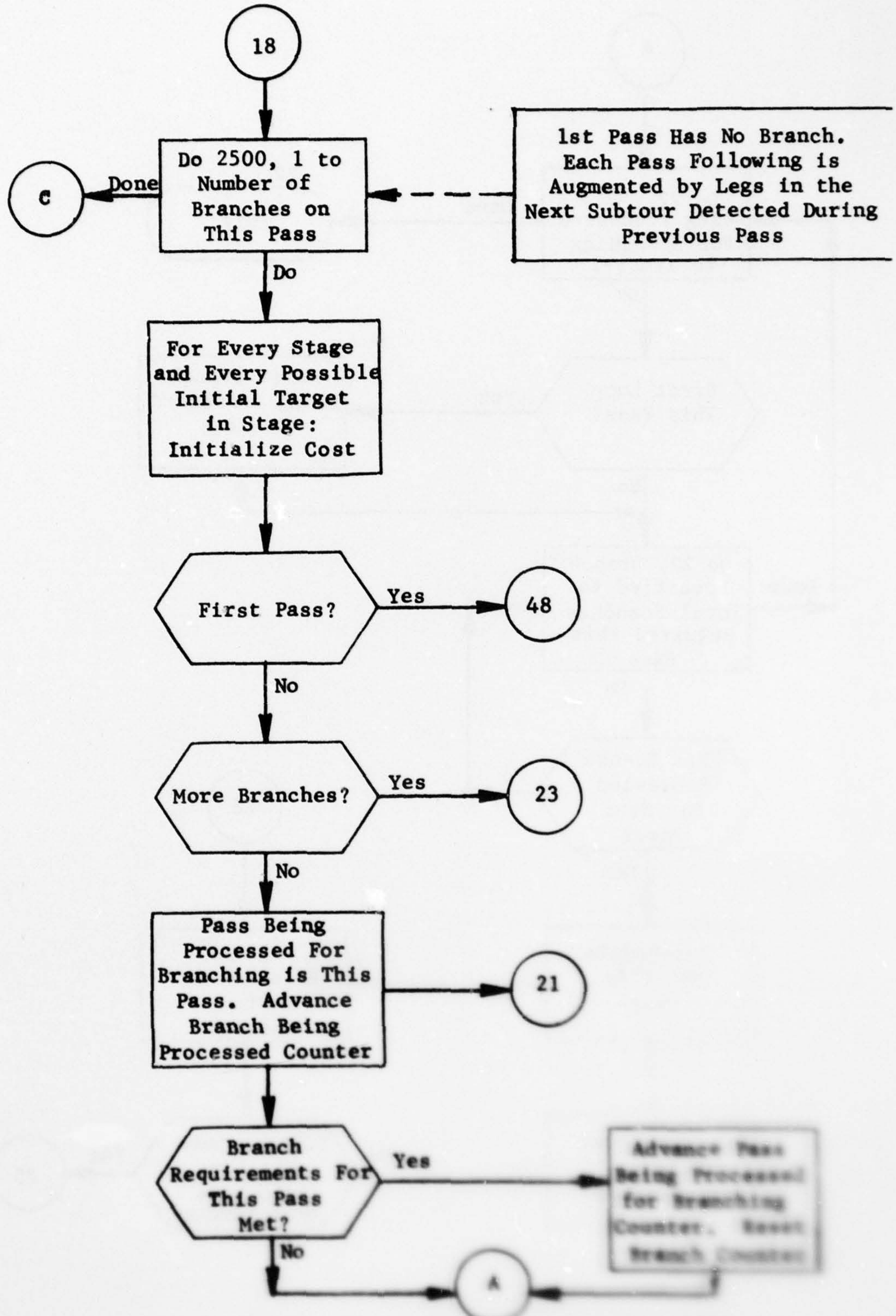
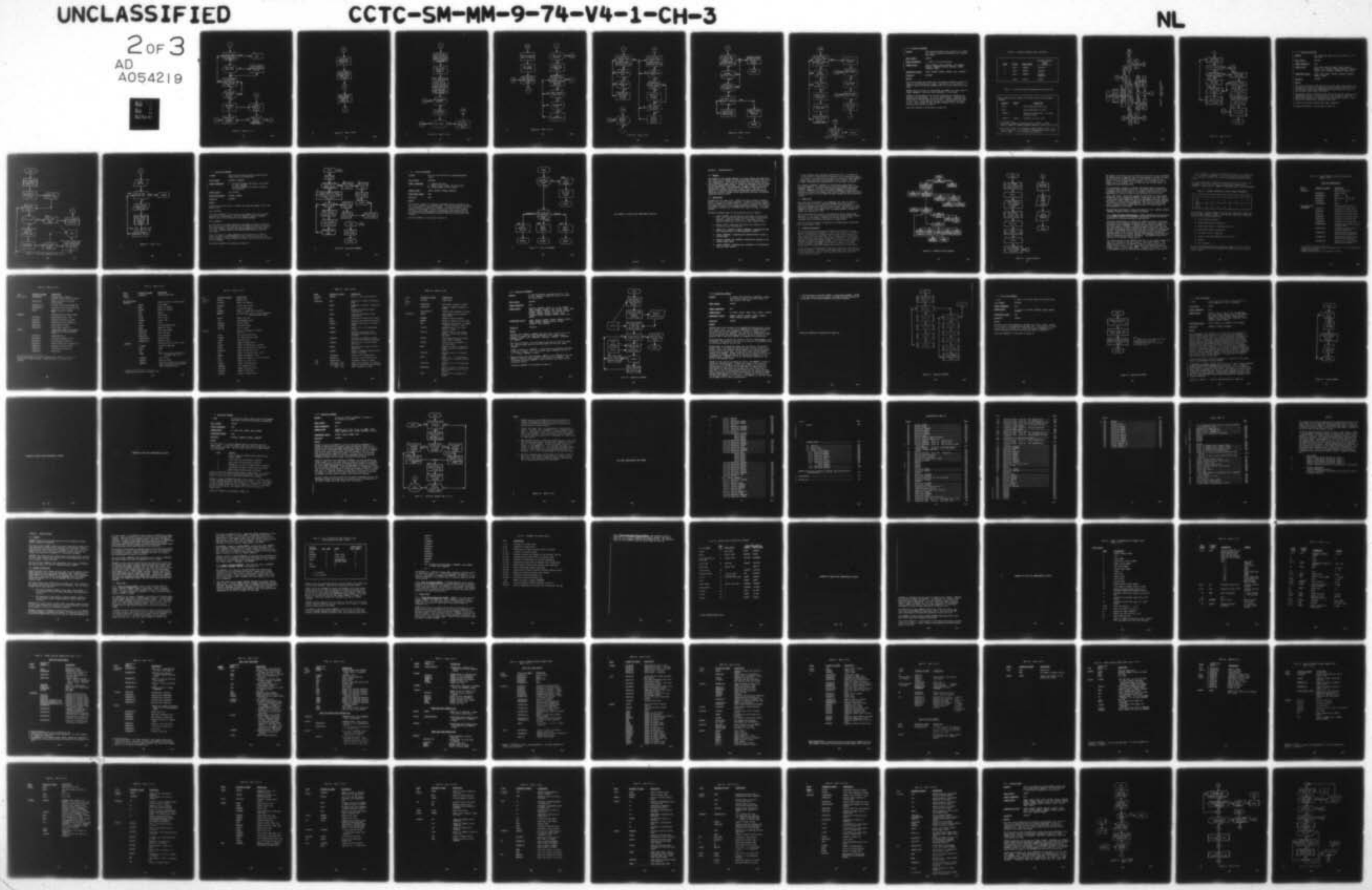


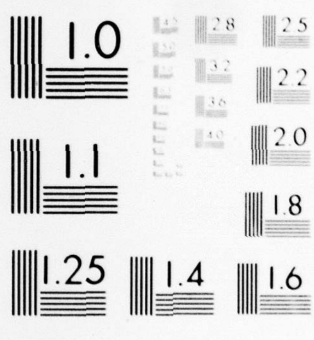
Figure 19. Branch and Bound Algorithm (Part 1 of 8)

AD-A054 219 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 15/7
THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK). PROG--ETC(U)
APR 78

UNCLASSIFIED CCTC-SM-MM-9-74-V4-1-CH-3 NL

2 OF 3
AD
A054219





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

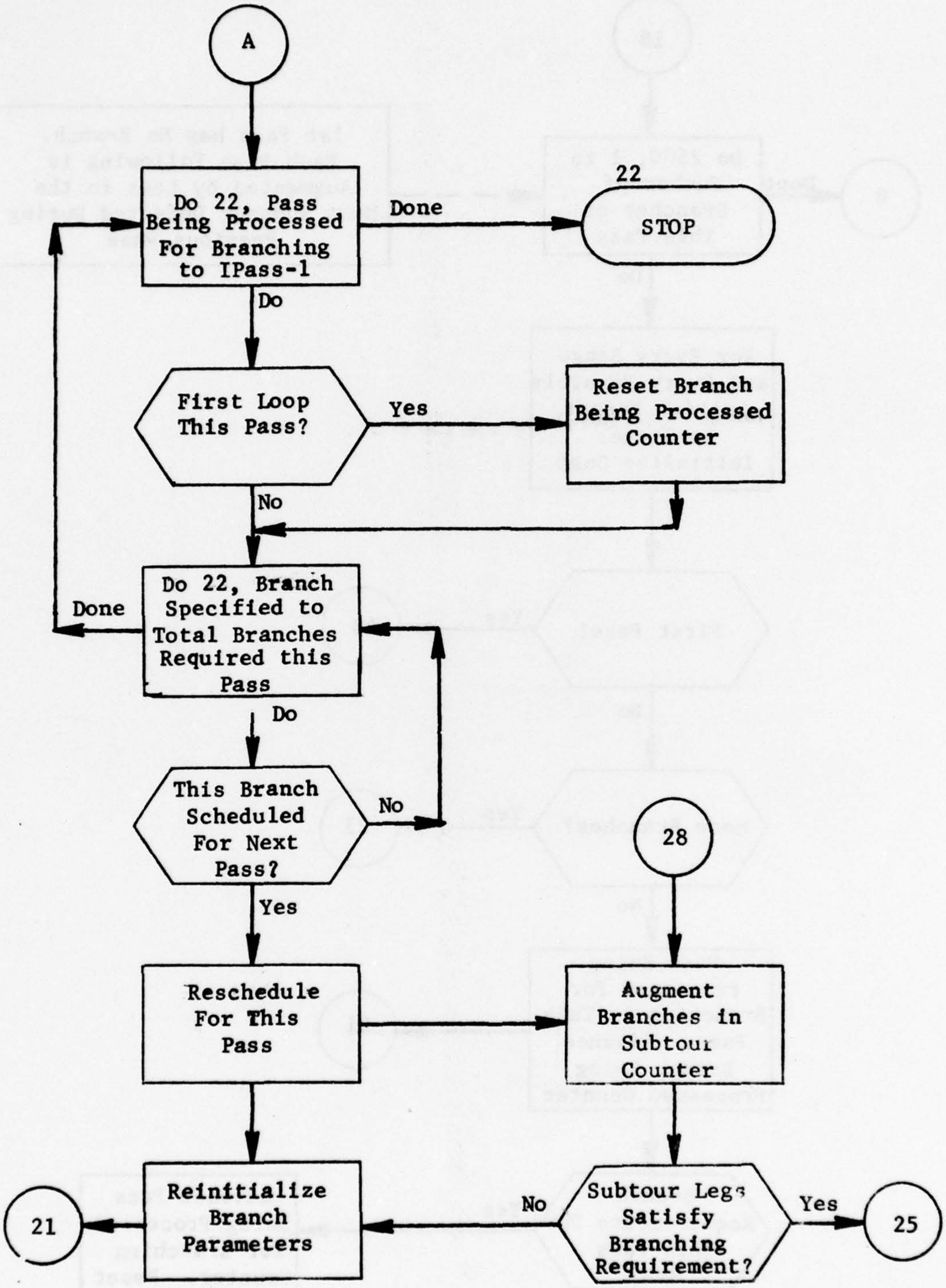


Figure 19. (Part 2 of 8)

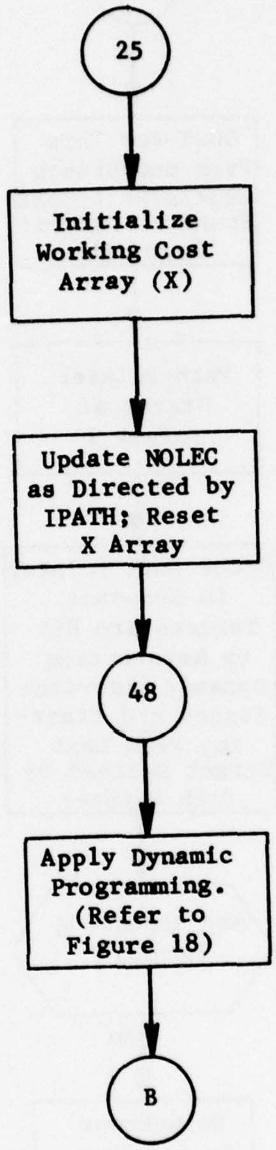


Figure 19. (Part 3 of 8)

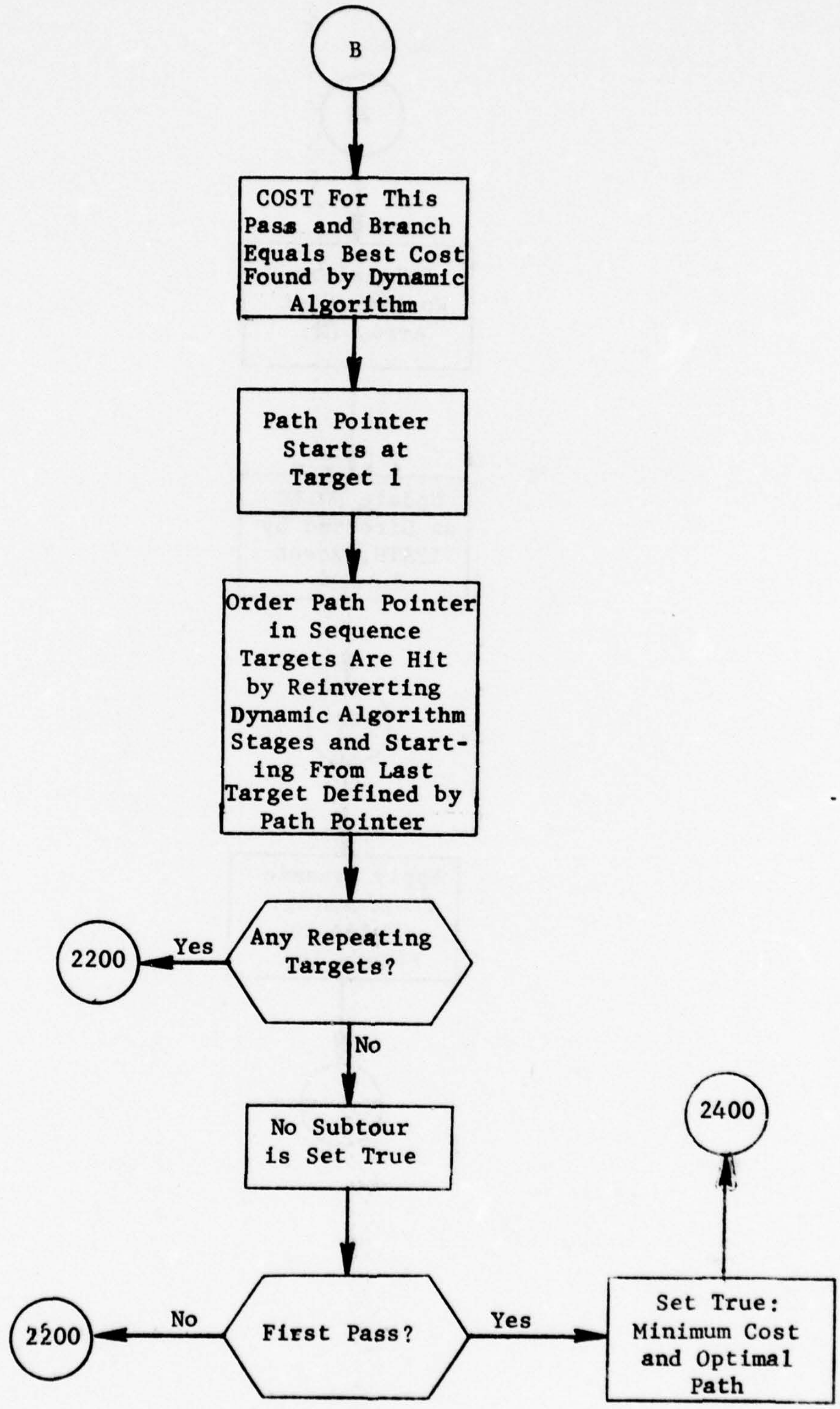


Figure 19. (Part 4 of 8)

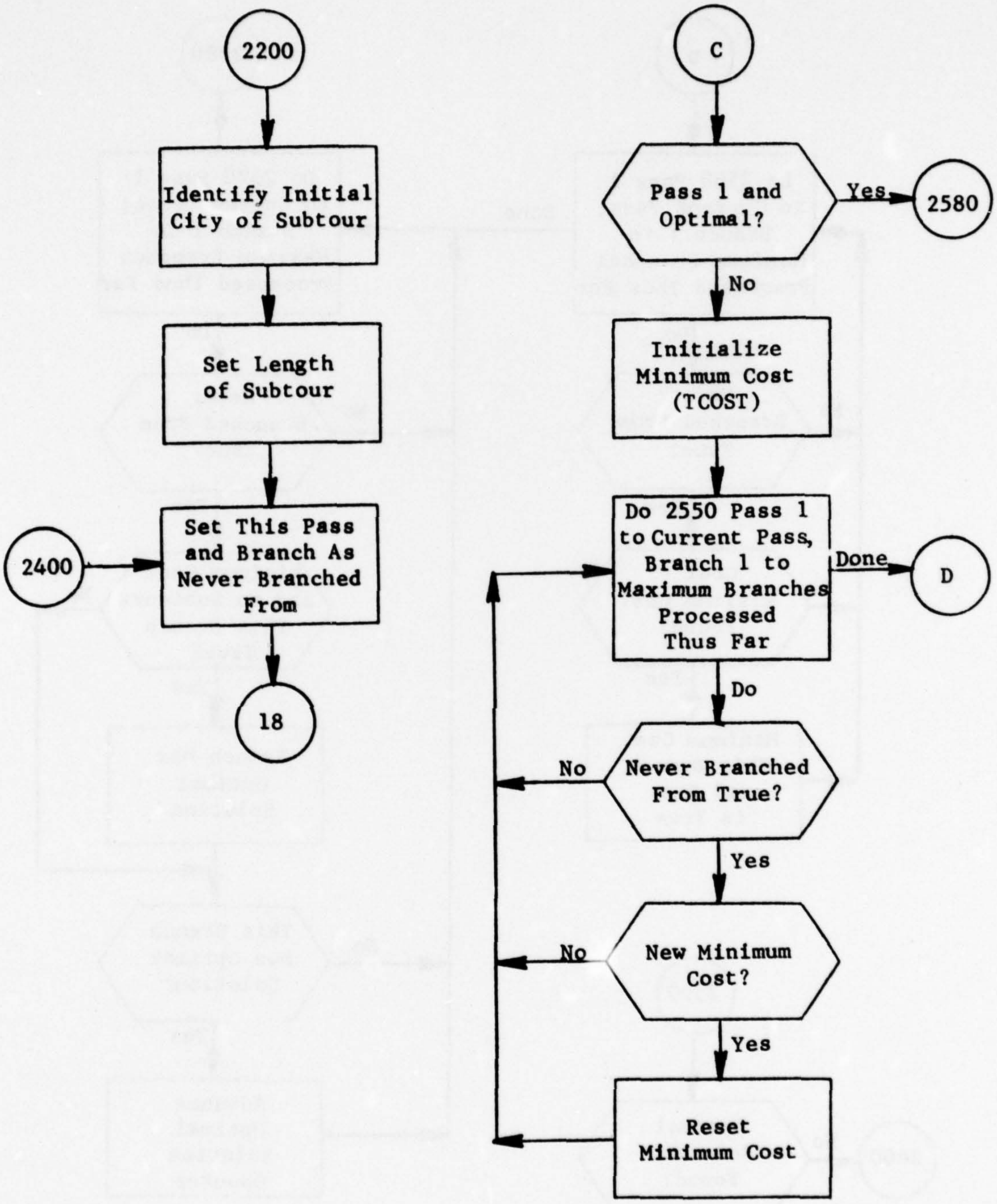


Figure 19. (Part 5 of 8)

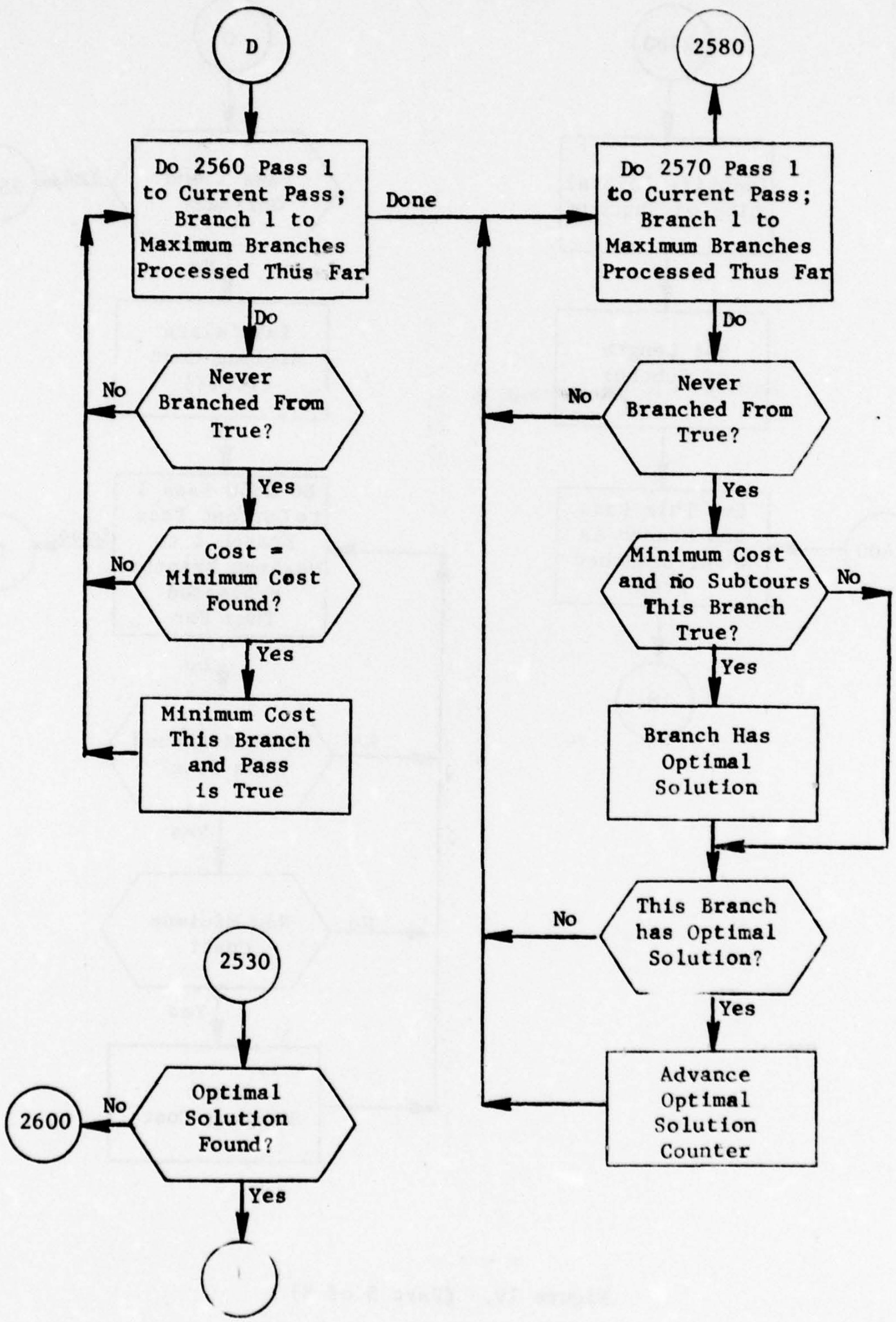


Figure 19. (Part 6 of 8)

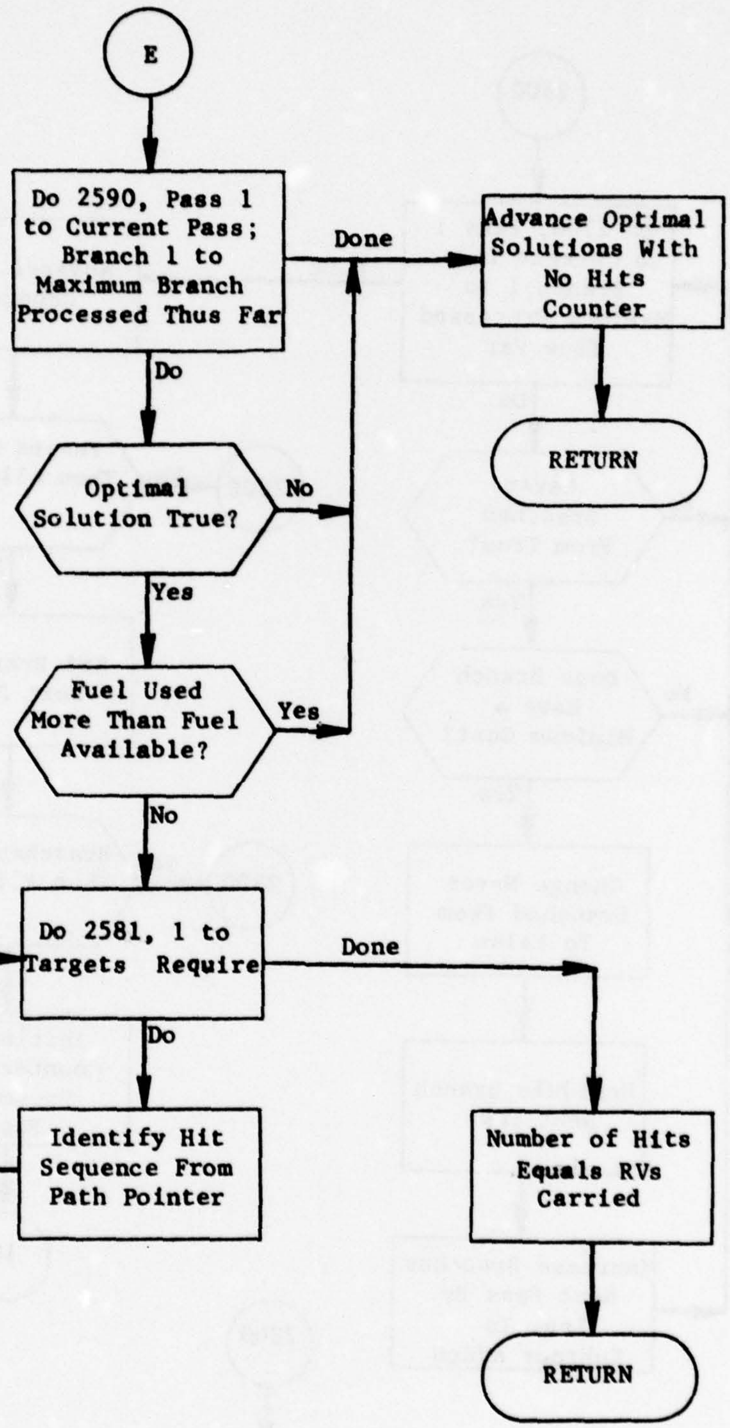


Figure 19. (Part 7 of 8)

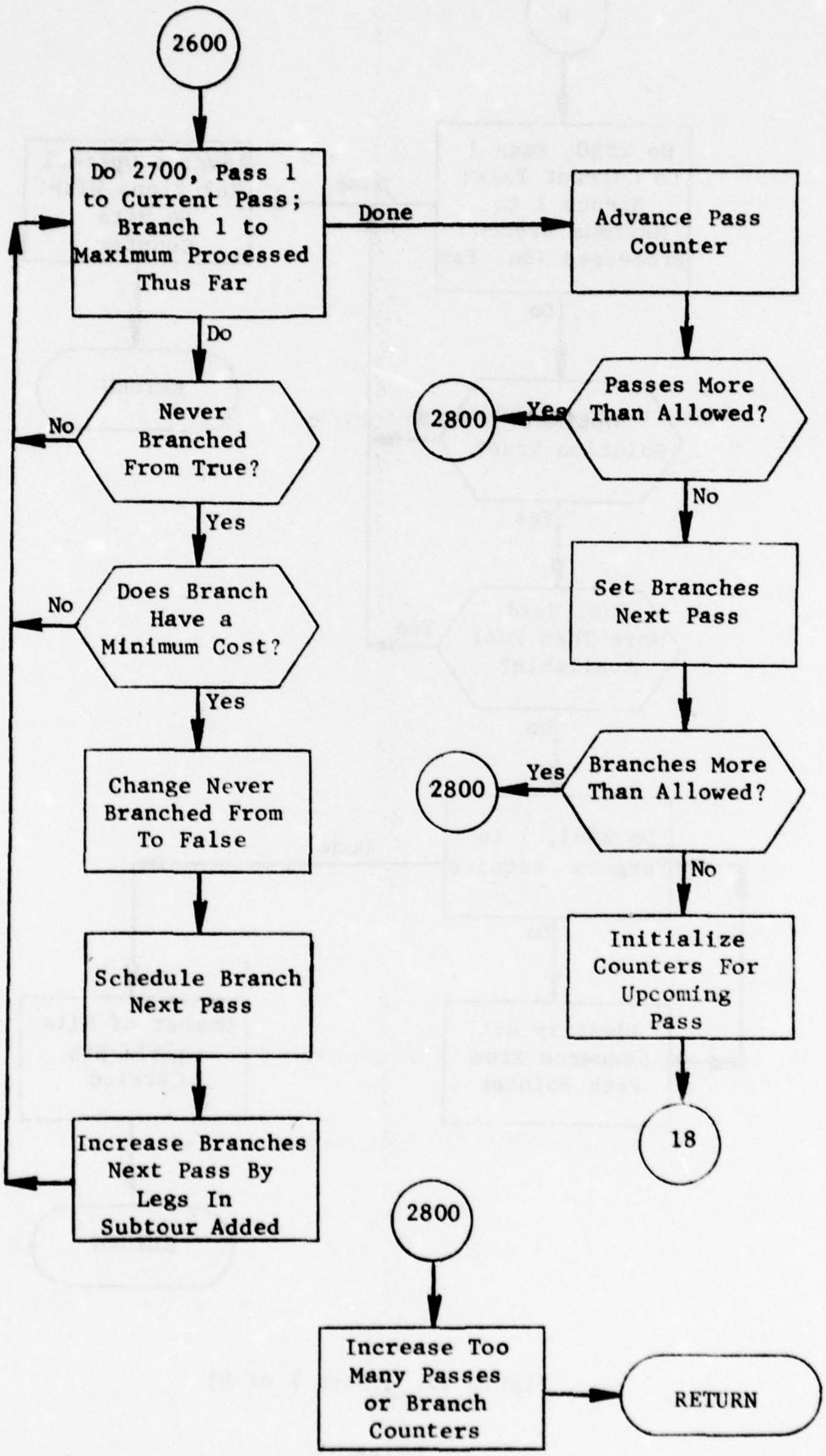


Figure 19. (Part 8 of 8)

2.6.7 Subroutine SETDATA

PURPOSE: This routine retrieves from a scratch file (ITABL) the correct footprinting parameters for a particular system.

ENTRY POINTS: SETDATA

FORMAL PARAMETERS: I - Index to the system desired

COMMON BLOCKS: CSYS4, FILABEL, FILES, FOOTDATA, ITP, MYIDENT, MYLABEL, PARAMETER, PENADD, SHRTDATA, SYSMAX, TSCRATCH, TWORD

SUBROUTINES CALLED: ABORT, RDARRAY, RDWORD, SETREAD, SKIP, TERMTAPE

CALLED BY: FOOTPRNT

Method:

This routine merely moves data from the footprint parameter scratch file, ITABL, to the footprint test arrays. (See table 8.) As subroutine TABLINPT reads the footprint parameter data, it writes them on the ITABL file.

SETDATA first retrieves the system MTYPE and IHNAME from those arrays in common /PARAMETR/, using the formal parameter I, as an index.

SETDATA then determines if the correct system data are already in the footprint testing storage. If so, the routine exists. Otherwise the ITABL file is searched for the correct data. (Table 9 shows the format of this file.) If the data are not found, the filehandler will abort the run. Upon finding the data, SETDATA transfers them to the appropriate array as listed in table 8.

Subroutine SETDATA is illustrated in figure 20.

Table 8. Footprint Parameter Data Transmission

MTYPE	SYSTEM	ARRAY LENGTH	FOOTPRINT TESTING COMMON BLOCK
1	Type-1	LLNGDAT	/FOOTDATA/
2	Type-2	LSHTDAT	/SHRTDAT/
3	Type-3	LPENDAT	/PENADD/ /FOOTDATA/
4	Type-4	LDSYS4	/CSYS4/

Table 9. Format for Footprint Parameter Data Scratch File

Each unique system is output on the ITABL file in the following format:

VARIABLE	LENGTH	DESCRIPTION
MTYPE	1	MIRV system functional type
IDATA	1	MIRV system data set number
"LENGTH"*	1	Length of footprint parameter table for this system
"TABLE"**	LENGTH	Footprint parameter table

*For MTYPE=1, LENGTH is LLNGDAT; MTYPE=2, LENGTH is LSHTDAT; MTYPE=3, LENGTH is LPENDAT; MTYPE=4, LENGTH is LDSYS4 (see table 8).

**For MTYPE=1, TABLE is the FOOTDATA common; MTYPE=2, TABLE is the SHRTDAT common; MTYPE=3, TABLE is the PENADD and FOOTDATA commons; MTYPE=4, TABLE is the CSYS4 common (see table 8).

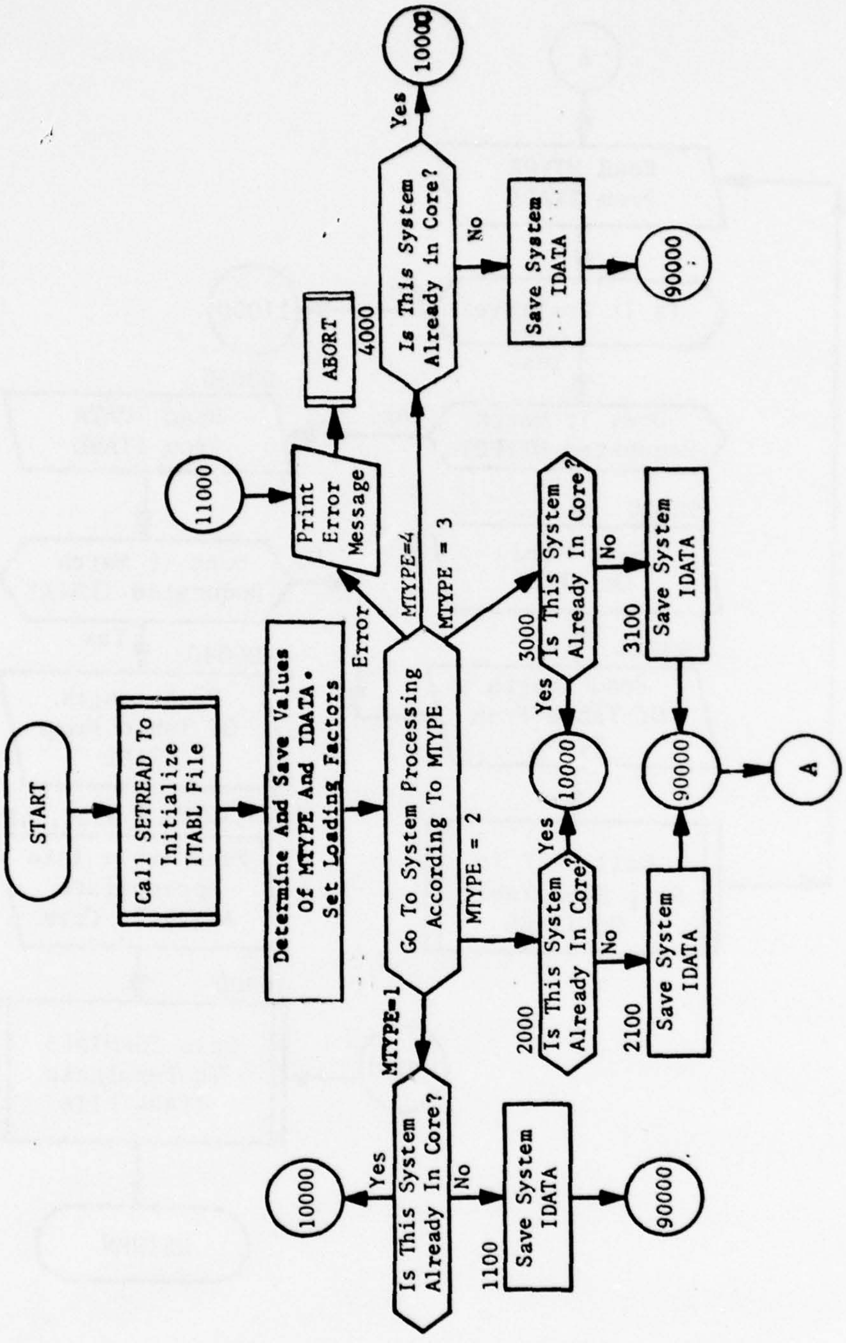


Figure 20. Subroutine SETDATA (Part 1 of 2)

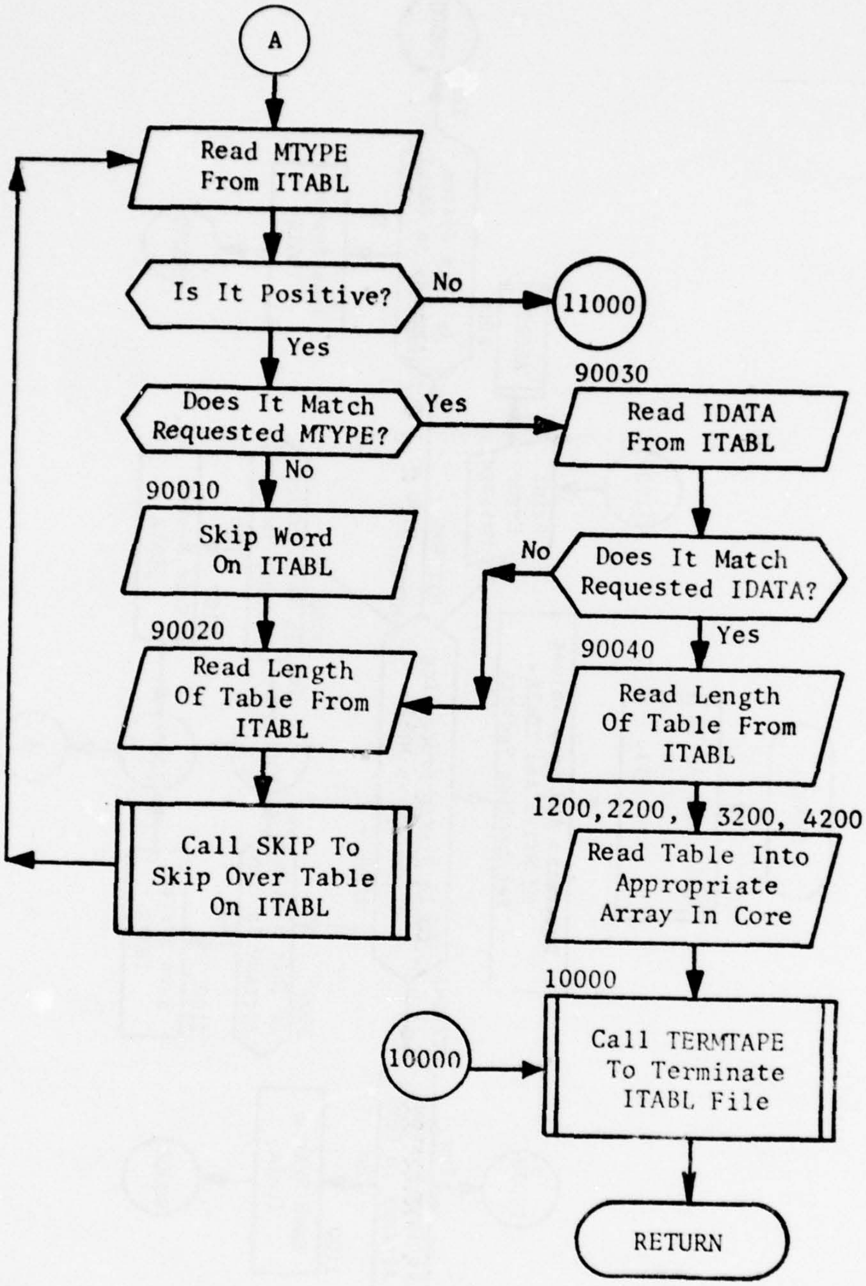


Figure 20. (Part 2 of 2)

2.6.8 Subroutine TABLINPT

PURPOSE:	This subroutine reads, prints, and saves all user input data.
ENTRY POINTS:	TABLINPT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	AXIS, CSYS4, EARTH, FILABEL, FILES, FOOTDATA, IFTPRNT, ITP, MYIDENT, MYLABEL, NOPRINT, PARAMETER, PENADD, PRINT, SHRTDATA, TRSCRATCH, TWORD, WAROUT
SUBROUTINES CALLED:	ABORT, AXES, NUMGET, SETDATA, SETWRITE, TERMTAPE, WRARRAY, WRWORD
CALLED BY:	FOOTPRNT

Method:

This routine initially reads and stores the user input print requests and following that continues to read the footprint parameter tables and stores the data into common block /PARAMETER/ and outputs the data on the footprint parameter data scratch file ITABL.

Reading and storing of parameter cards follows the outline as given in the Users Manual, Volume IV. A system card is read followed by cards that define the constants used for describing the given MIRV system. A blank system card terminates the reading of input data.

A print of all inputs follows after input completion.

Subroutine TABLINPT is illustrated in figure 21.

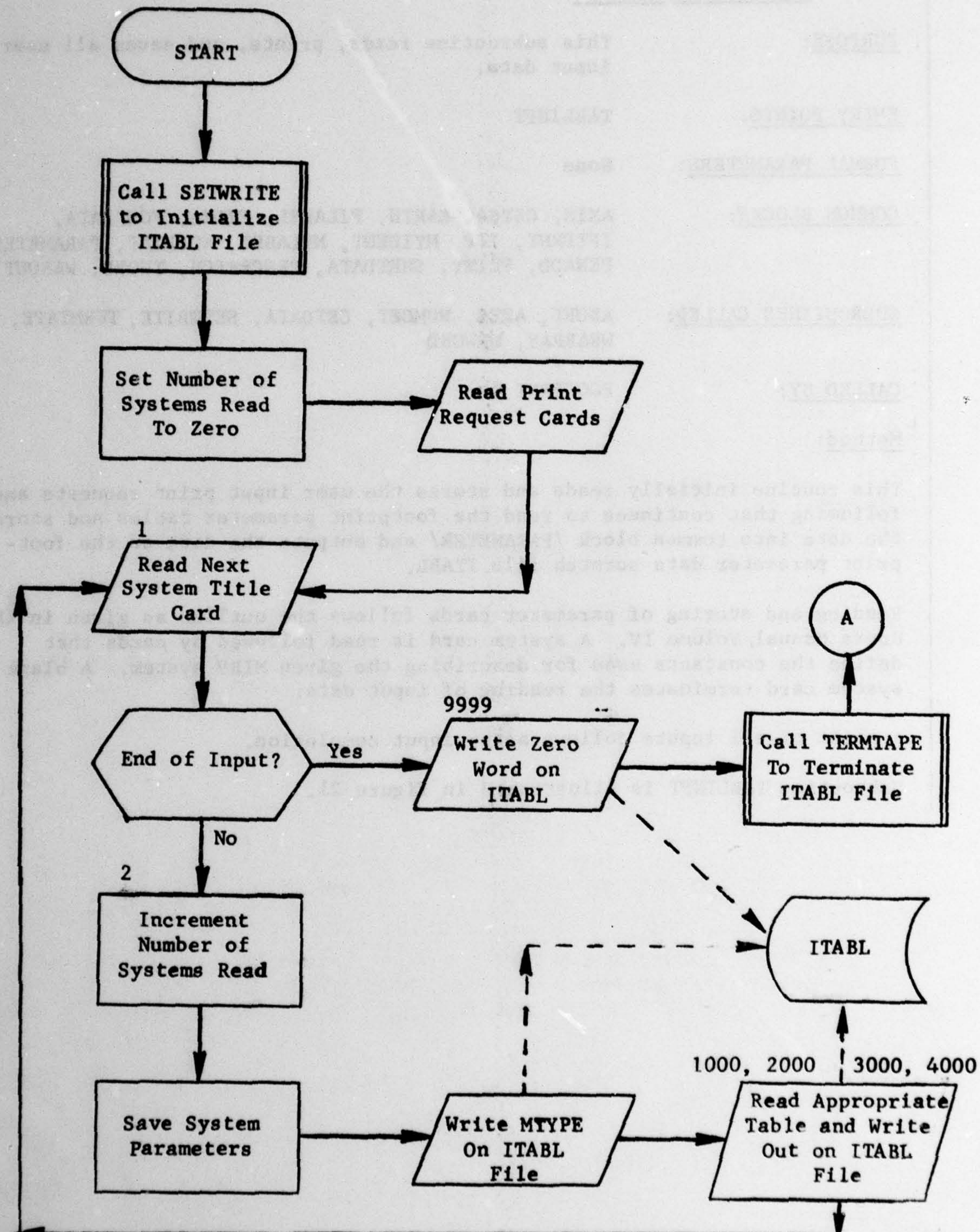


Figure 21. Subroutine TABLINPT (Part 1 of 2)

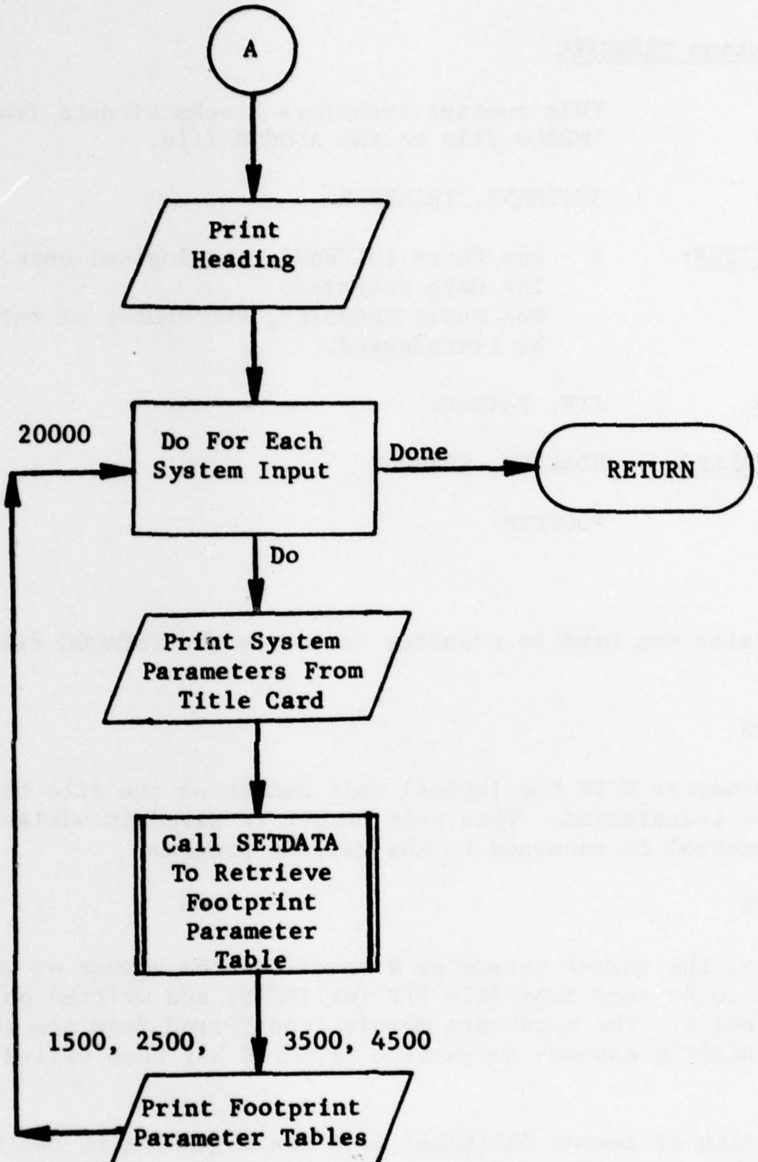


Figure 21. (Part 2 of 2)

?.6.9 Subroutine TRANSFER

PURPOSE: This routine transfers blocks of data from the TMPALO file to the ALOCGR file.

ENTRY POINTS: INITRANS, TRANSFER

FORMAL PARAMETERS: N - For Entry INITRANS, the logical unit number for data transfer.
For Entry TRANSFER, the number of words to be transferred.

COMMON BLOCKS: ITP, RAIDATA

SUBROUTINES CALLED: RDARRAY, WRARRAY

CALLED BY: FOOTPRNT

Method:

These two entries are used to transfer data from the TMPALOC file to the ALOCGRP file.

Entry INITRANS

The formal parameter N is the logical unit number of the file to which data are to be transferred. This unit number is saved in variable IWRITE, and control is returned to the calling program.

Entry TRANSFER

For this entry, the formal parameter N specifies the number of words of data that are to be read from file ITP (or IREAD) and written on logical file number IWRITE. The words are merely transferred from one tape to the other. TRANSFER assumes subroutine SETWRITE has been called for file IWRITE.

Note: The length of common /RAIDATA/ from the beginning to LRAID is stored in LRAID. Since TRANSFER uses this length in determining the size of temporary storage, changes in common /RAIDATA/ should be reflected in this variable.

Subroutine TRANSFER is illustrated in figure 22.

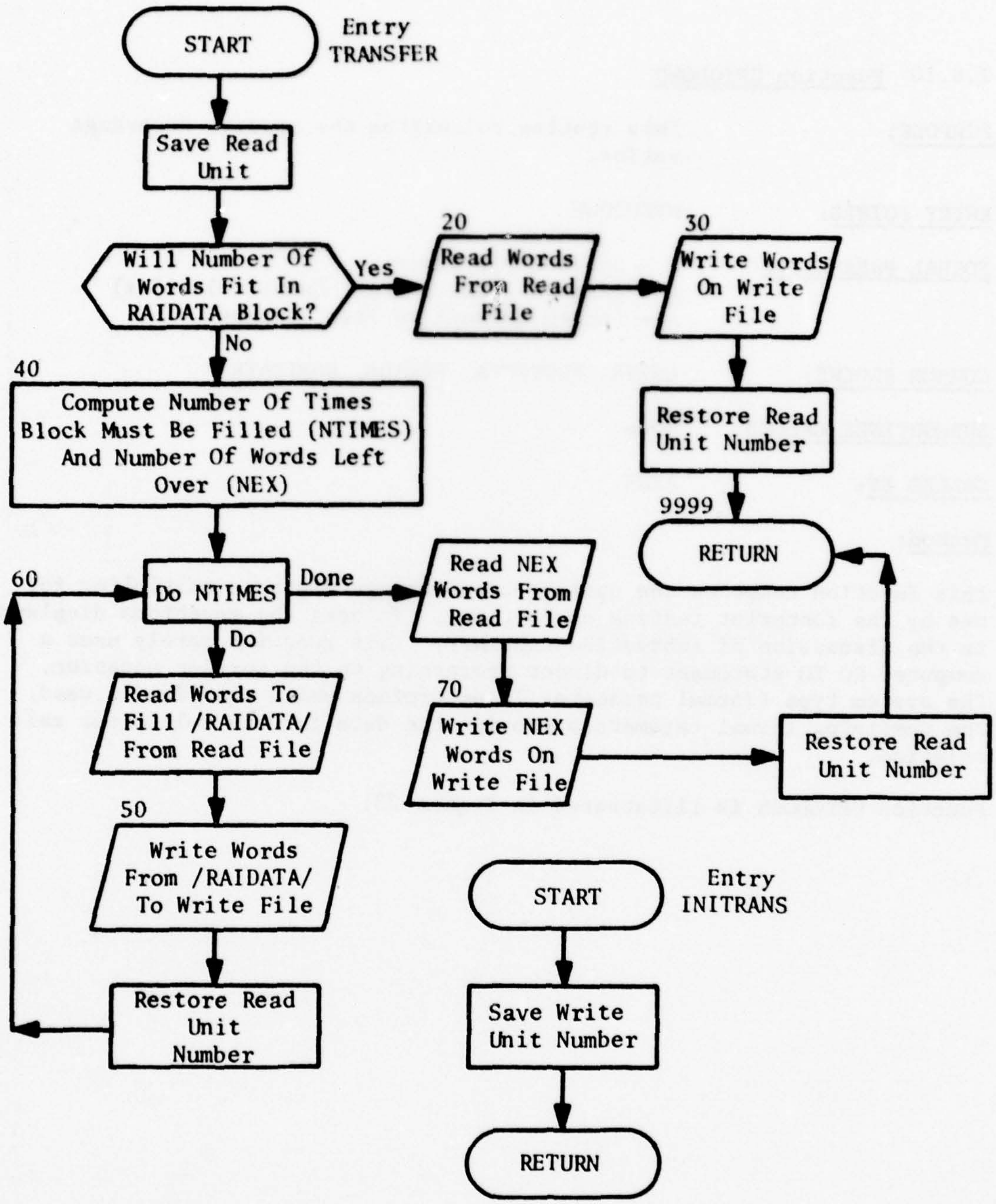


Figure 22. Subroutine TRANSFER

?.6.10 Function UPTODOWN

PURPOSE:	This routine calculates the uprange-downrange ratios.
ENTRY POINTS:	UPTODOWN
FORMAL PARAMETERS:	I - System type index R - Range to first target (nautical miles) AZ- Launch azimuth to first target
COMMON BLOCKS:	CSYS4, FOOTDATA, PENADD, SHRTDATA
SUBROUTINES CALLED:	None
CALLED BY:	AXES

Method:

This function computes the uprange-to-downrange distance multiplier for use by the footprint testing subroutines. It uses the equations displayed in the discussion of subroutine TABLINPT. This function merely uses a computed GO TO statement to direct processing to the correct equation. The system type (formal parameter I0 determines which equation is used. The remaining formal parameters provide the data for the multiplier calculation.

Function UPTODOWN is illustrated in figure 23.

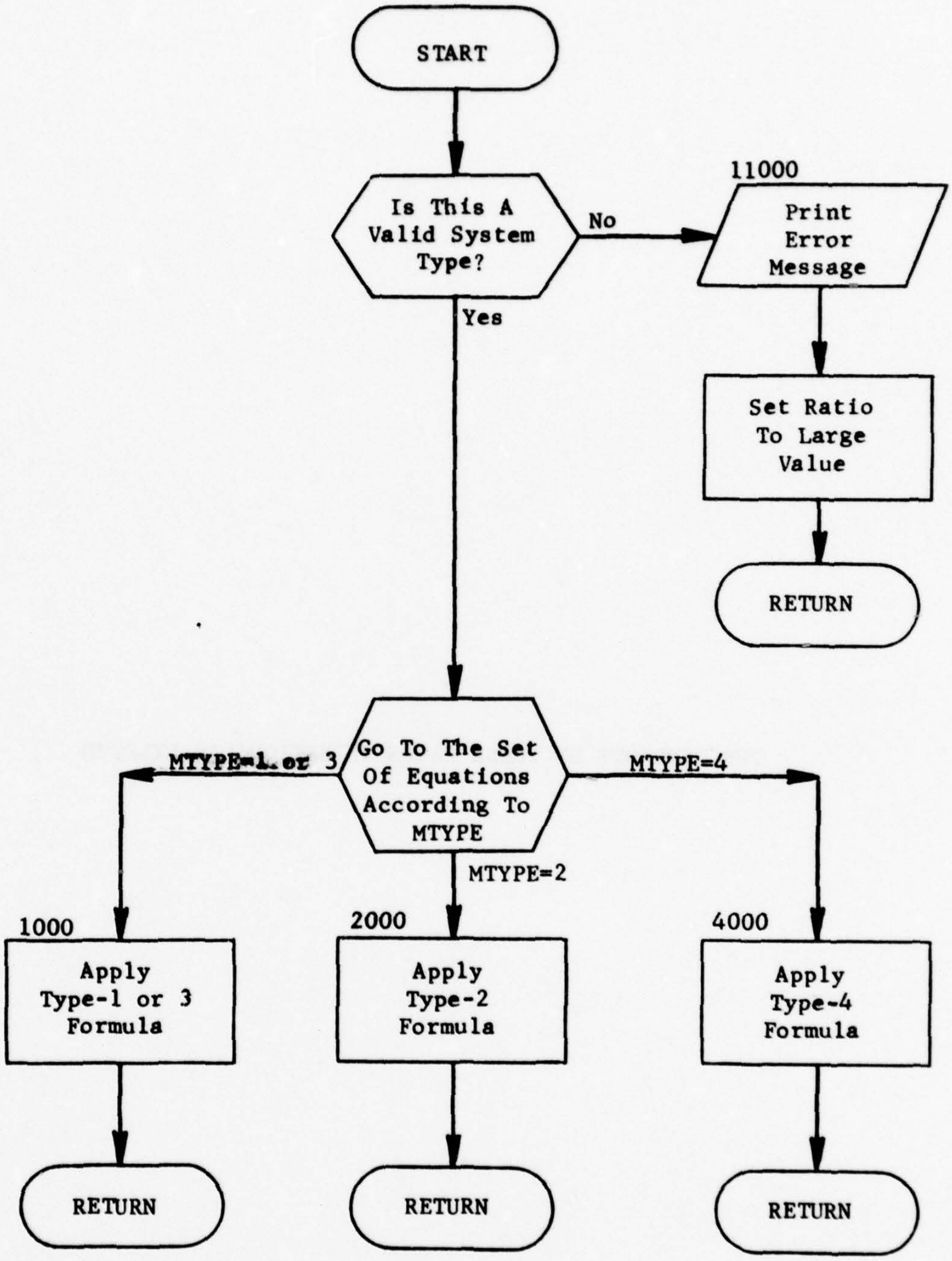


Figure 23. Function UPTODOWN

THE CONTENTS OF THESE PAGES INTENTIONALLY DELETED

SECTION 3. PROGRAM POSTALOC

3.1 Purpose

The purpose of the program POSTALOC is to write missile and bomber delivery plans from the weapon-to-target allocations developed by the allocator, program ALOC. In the case of missiles, this is a simple process since the missile flight plans are completely determined once the target and launch coordinates are known. In the case of bombers, the process is more complicated. The development of bomber sorties requires the association of several strikes in a single sortie. Moreover, it is necessary to associate each sortie with specific launch and recovery bases and to select a flight profile which specifies where low-altitude capability should be used.

3.2 Input Files

POSTALOC uses two input files: BASFILE and either ALOCGRP or TMPALOC. BASFILE is written by program PREPALOC. If MIRVs are present, POSTALOC uses the ALOCGRP file which contains the target allocation from ALOC, as rearranged and rewritten by programs ALOCOUT and FOOTPRNT. If no MIRVs are present, POSTALOC uses the TMPALOC file which is output by program ALOCOUT.

Subroutine GETGROUP reads the following data from the BASFILE:

a. Common /MASTER/, containing basic information about the data base, such as number of weapon groups, number of penetration corridors, number of targets, etc. (See table 13 for a complete description of this and all other common blocks.)

b. Common /FILES/, containing the logical unit numbers of all the files in the Plan Generator.

c. Common /C9/, previously common /CORRCHAR/, containing the general characteristics of each penetration corridor.

d. Common /ASMTABLE/, containing the characteristics of each of the ASM types.

e. Commons /PAYLOAD/ and /PAYDATA/, defining the payloads of the various bomber types.

f. Common /DPENREF/, containing the coordinates of the depenetration and refuel points.

Subroutine GETGROUP skips subsequent BASFILE data until it reaches the word of Hollerith Z's which marks the beginning of the weapon group data. It then reads for each weapon group common /C7/, previously common /GRPDATA/, containing basic data about the weapon group and its bases, and common /GRPTYPE/, containing type characteristics of the bomber or missile.

The reading of ALOCGRP or TMPALOC for missile groups takes place in subroutine MISASGN. For bomber groups, subroutine PRERAID reads common /STRKSUM/, which is a summary of the weapon allocation by penetration corridor. PRERAID makes a call on subroutine GENRAID to process each penetration corridor, and GENRAID reads from the ALOCGRP common /C3/, which contains the data on all the targets assigned through the given penetration corridor.

3.3 Output File

The output file for POSTALOC is the STRKFILE, the format for which is shown in tables 10 and 11. Subroutine OUTSRT writes one record for each bomber sortie, describing the sortie plan and characteristics, and subroutine MISASGN writes the missile event plans. The first word on the STRKFILE is user's input TARFAC (used in program PLANOUT); missile and bomber plan immediately follow.

The file contains one record for each bomber and missile flight plan generated. In the case of bombers which refuel, two records are present: the first for the primary, refueled plan and the second for the alternate plan to be used in the event of a refuel abort.

The end of data on the file is signalled by a dummy bomber record which has a group number of 999.

3.4 Concept of Operation

The sortie definitions developed by POSTALOC are generated by weapon groups, one penetration corridor at a time. They consist of an ordered list of the targets to be struck by each sortie, a specification of which targets to strike with ASMs, and an estimate of the low-altitude range allotted for use before, versus after, the first target (and in any legs preceding the corridor origin). The sortie definition does not include the actual coordinates for the events. Thus for the bomber events it remains for PLANOUT to add these coordinates, calculate release points for ASMs, and compute time of entry into defense zones.

Figure 38 shows the relationship among the various major subroutines in the post-allocator. The arrows in the figure point from each subroutine to the subroutines it calls. Thus the arrows illustrate simply the calling sequence hierarchy.

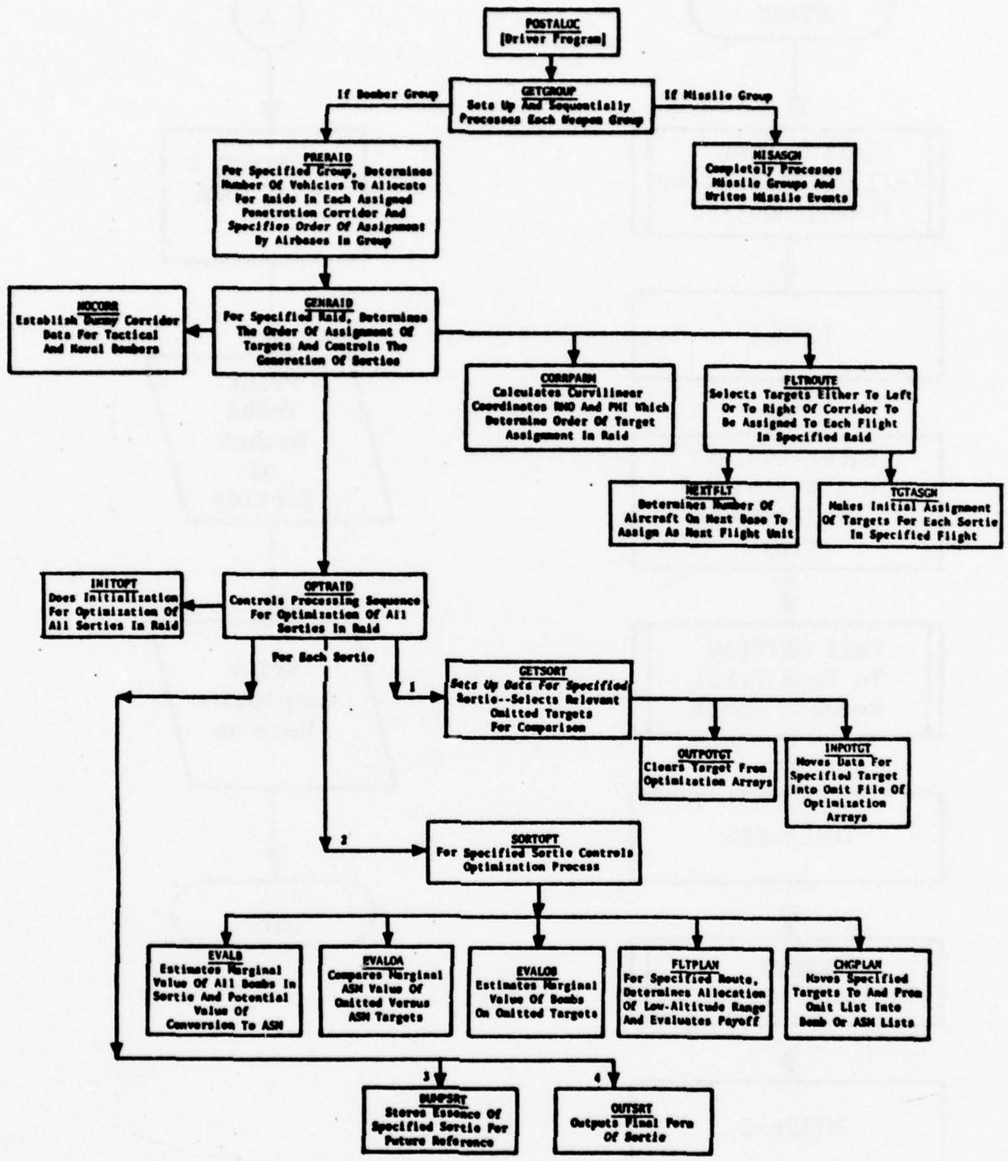


Figure 38. POSTALOC Calling Sequence

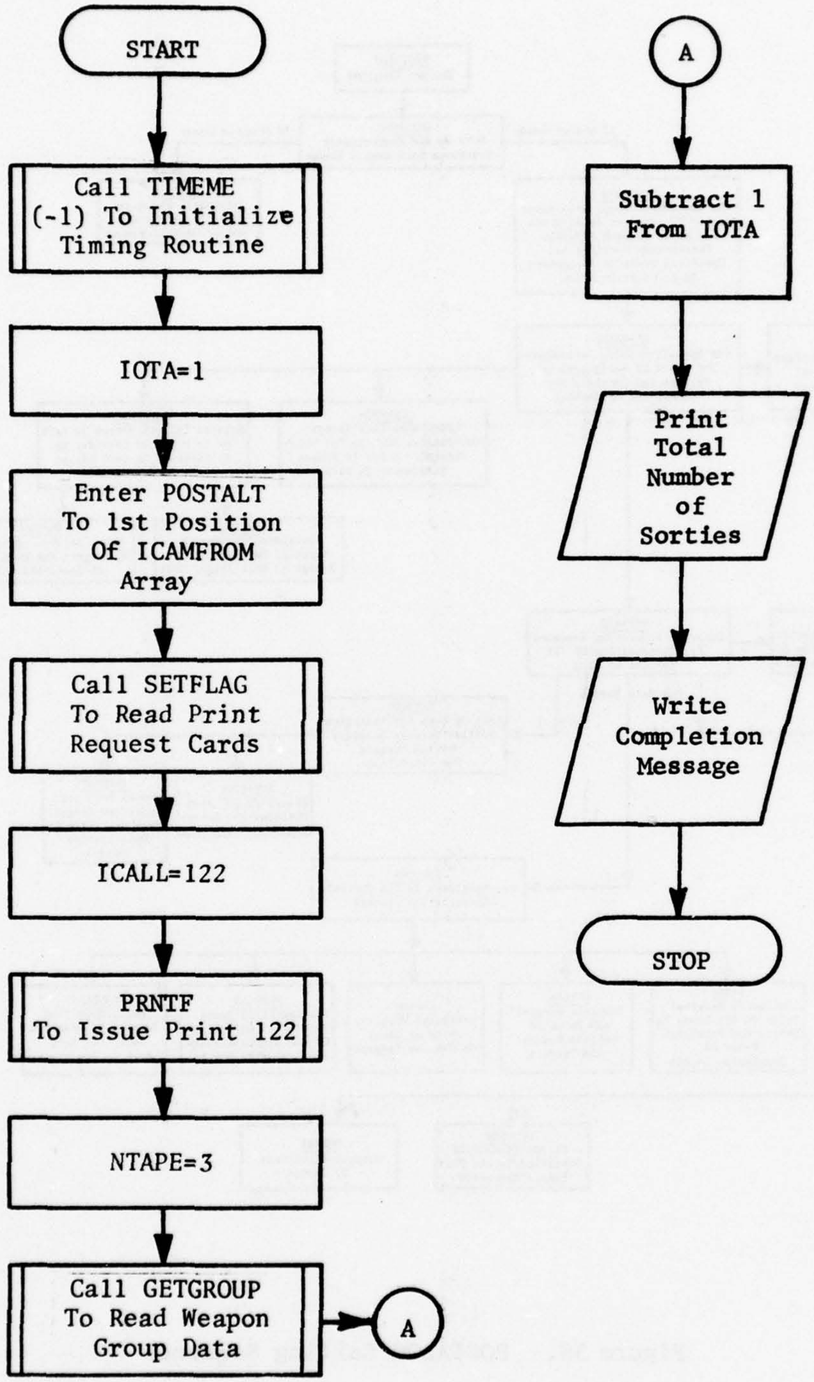


Figure 39. Program POSTALOC

the density of strikes on the two sides of the corridor is quite different, the flights going to opposite sides are kept roughly in balance by comparing the value of ϕ before deciding to which side to assign the next flight. In order to maintain this balance, it is desirable to have at least five or six flights. Thus if there are four or fewer bases, two flights are sent from each base.

If no penetration corridor is defined, the launch bases are processed in order of their absolute values of ϕ alternating from one side of the coordinate system to the other, in an attempt to make the sortie paths approximate as closely as possible the direction of the PHI lines.

Within each flight, strikes are assigned to one sortie at a time working through the list of unassigned strikes. Processing for the next sortie in the flight always begins with the first unassigned strike and continues from there. Strikes actually assigned to each sortie are always arranged in the sortie in order of increasing RHO. This gives the initial time order or sequence of the strikes which is used as a starting point for the optimization of the sortie.

The principal subroutines used for raid generation are: PRERAID, GENRAID, CORRPARM, FLTROUTE, NEXTFLT, TGTASGN, and NOCORR.

3.4.2 Setup for Sortie Optimization. Before describing the optimization of individual sorties, it is necessary to describe briefly the way the information is structured during the optimization.

During the optimization of a sortie, all targets relevant to the sortie are entered into a detailed computation array. This array (common /C1/, previously common /SORTYTGT/ -- targets for this sortie) includes not only the targets or strikes originally assigned to the sortie but also any targets omitted by prior sorties that may be relevant as target alternatives. The SORTYTGT arrays include not only the index to the targets in the basic target list, but also the coordinates, value, and defense characteristics for the targets together with temporary scratch-pad data, used in estimating the value of the sortie. The SORTYTGT arrays have capacity for 25 separate target entries.

The index positions in the SORTYTGT array that do not contain targets are said to be "available" and are listed in a file named "IAVAIL." The remaining positions in the SORTYTGT array will contain points of the present sortie listed in a file named "IHIT" and possible alternative targets listed in a file named "IOMIT." Actually, positions 1, 2, and 3 of the SORTYTGT array are reserved for nontarget points in the sortie.

ition 1 (IORIG = 1) is used to represent the origin of the penetration idor. Position 2 (IRECOVER = 2) is used to represent the recovery point. Position 3 (IDITCH = 3) is used to represent termination of the mission.

These conventions make it possible to define a sortie with a simple list of numbers. This sortie definition is contained in common block /CURSORTY/ (current sortie). **Table 12 illustrates such a sortie definition.**

Table 12. Example Definition of Sortie (Common /CURSORTY/)

	1	2	3	4	5	6	7	8
IFLY	1	7	7	8	5	5	2	3
IHIT	1	7	-9	8	-5	-4	2	3
IOMIT	6	13						
IAVAIL	10	11	12	15	14			

By convention, a negative number in the IHIT list indicates an ASM, and a positive number indicates a bomb. Thus the IFLY, IHIT table illustrated represents the following operations:

a. Leave origin of corridor

b. Bomb target listed in position 7

c. From the vicinity of 7 launch an ASM at 9

d. Bomb target listed in position 8

e. Fly near 5 to hit 5 with ASM

f. Strike 4 with ASM launched from vicinity of 5

g. Recover

h. End of mission.

The omit list indicates possible alternative targets listed in positions 6 and 13, while the avail list indicates five empty cells that could be used if needed.

Table 13. Program POSTALOC External Common Blocks
(Part 1 of 9)

INPUT DATA FROM BASFILE

BLOCK**	VARIABLE OR ARRAY*	DESCRIPTION
/ASMTABLE/		ASM characteristics
	IWHDASM(20)	Warhead index (for ASMs)
	RANGEASM(20)	Range (for ASMs)
	RELASM(20)	Reliability (for ASMs)
	CEPASM(20)	CEP (for ASMs)
	SPEEDASM(20)	Speed (for ASMs)
/C9/ (previously /CORRCHAR/)		Corridor characteristics
	PCLAT(30)	Latitude of corridor point
	PCLONG(30)	Longitude of corridor point
	RPLAT(30)	Latitude of corridor origin
	RPLONG(30)	Longitude of corridor origin
	ENTLAT(30)	Latitude of corridor entry
	ENTLONG(30)	Longitude of corridor entry
	CRLENGTH(30)	Distance from corridor entry to corridor origin
	KORSTYLE(30)	Parameter to adjust mode of corridor penetration
	ATTRCORR(30)	High-altitude attrition per nautical mile unsuppressed
	ATTRSUPP(30)	High-altitude attrition per nautical mile suppressed
	HILOATTR(30)	Ratio low- to high-altitude attrition (less than 1)

*Parenthetical values indicate array dimensions. All other elements are single word variables.

**Ordered alphabetically, not by position in core.

Table 13. (Part 2 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/C9/ (cont.)	DEFRANGE(30)	Characteristic range of corridor defense (nautical miles)
	NPRCRDEF(30)	Number of attrition sections this corridor
	DEFDIST(30,3)	Distance of each precorridor leg
	ATTRPRE(30,3)	Attrition in each precorridor leg
	NDATA	Number of words in common /C9/
/DPENREF/		Depenetration and refuel points
	DPLINK(50)	Depenetration point link
	DPLAT(50)	Depenetration point latitude
	DPLONG(50)	Depenetration point longitude
	REFLAT(20)	Refuel point latitude
	REFLONG(20)	Refuel point longitude
/FILES/		Logical unit number and maximum length for all Plan Generator files
	TGTFILE(2)*	Target data file
	BASFILE(2)	Data base information file
	MSLTIME(2)	Fixed missile timing file
	ALOCTAR(2)	Weapon allocation by targets file
	TMPALOC(2)	Temporary allocation file
	ALOCGRP(2)	Allocation by group file
	STRKFIL(2)	Strike file

*In two-word arrays, first word is logical unit number; second word is maximum file length in words. Single variables are logical unit numbers.

Table 13. (Part 3 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/FILES/ (cont.)	PLANTAPE*	Detailed plans tape
/C7/ (previously /GRPDATA/)		Characteristics of weapon groups
	IGROUP	Group number
	NWPNS	Number of weapons
	NVEHGRP	Number of vehicles
	IREG	Region
	ITYPE	Weapon type
	IALERT	Alert status
	IREFUEL	Refuel code
	YIELD	Yield
	ISTART	Starting weapon index
	NBASE	Number of bases
	IBASE(150)	Base index number
	BLAT(150)	Base latitude
	BLONG(150)	Base longitude
	IPAYLOAD(150)	Payload index
	VONBASE(150)	Number on base
/GRPTYPE/		Characteristics of weapon types
	ISIMTYPE	Hollerith type name
	RANGE	Range
	CEP	CEP
	SPEED	Speed (For missiles replaced by minimum flight time)
	ALERTDLY	Alert delay
	NALRTDLY	Nonalert delay
	RANGEDEC	High/low altitude fuel consumptio ratio (For missiles replaced by time of flight factor)

* These files are output on magnetic tape.

Table 13. (Part 4 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/GRPTYPE/ (cont.)	ICLASS	Weapon class
	NOPERSQN	Number per squadron
	LCHINTVL	Launch time interval
	SPDLO	Low-altitude speed
	SIMLUNCH	Number of simultaneous launches
	RANGREF	Refueled range (for missiles replaced by minimum flight range)
	NMPSITE	Number per site
	IREP	Reprogramming index
	IRECMODE	Recovery mode
	IPENMODE	Penetration mode
	FUNCTION	Function code
/MASTER/		Run ID, and quantity of QUICK entities
	IHDATE	Date of run initiation
	IDENTNO	Run identification number
	ISIDEM	Attacking side
	NRTPT	Number of route points
	NCORRM	Number of penetration corridors
	NDPEN	Number of depenetration corridors
	NRECOVER	Number of recovery bases
	NREF	Number of directed refuel areas
	NBNDRY	Number of boundary points
	NREG	Number of command and control regions
	NTYPE	Number of weapon types
	NGROUP	Number of weapon groups
	NTOTBASE	Total number of bases
	NPAYLOAD	Number of payload types
	NASMTYPE	Number of ASM types
	NWHDTYPE	Number of warhead types

Table 15. (Part 2 of 16)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/CONTROL/ (cont.)	KWALSRT	Set to 1; not currently used
/CORRIDOR/	CLAT	Latitude of corridor orientation point
	CLONG	Longitude of corridor orientation point
	PLAT	Latitude of corridor origin (route point)
	PLONG	Longitude of corridor origin (route point)
	ELAT	Latitude of corridor entry point
	ELONG	Longitude of corridor entry point
	CLENGTH	Distance from entry to corridor origin
	KORPWR	Power of Y vs. X in calculation of PHI
	CORATTR	Attrition of corridor with unsuppressed defenses at high altitude
	CORSATTR	Attrition of corridor at high altitude with defenses suppressed
	ATTRHILO	Ratio of low- to high-altitude attrition
	RNGDEF	Characteristic range of defense operations
	NPREDEF	Number of separate defended zones prior to corridor
	DISTDEF(3)	Length of Ith defended zone
	PREATTR(3)	Attrition in Ith defended zone
/C4/	MYASGN(1030)	Assignment status of target
/C5/	JTGT(ISTRK, ISRT) with ISTRK = 10 ISRT = 100	Index (IT) to target for ISTRKth strike in ISRTth sortie in present raid; negative for ASMs

Table 15. (Part 3 of 16)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/C5/ (cont.)	DUDJ(30)	Dummy array
/C6/	MYBASE(100)	Base index assigned to sortie
	NASGN(100)	Number of targets assigned to sortie
	INDEXVEH(110)	Vehicle index assigned to sortie
/CURSORTY/	NUMHIT	Total targets hit by current sortie plan
	NUMBOMB	Number of targets bombed
	NUMASM	Number of targets hit with ASMs
	NHIT	Number of sortie points -- NUMHIT + (Origin, Recovery, Ditch)
	IFLY(13)	SORTYTGT index to Ith route point or target
	IHIT(13)	SORTYTGT index to Ith flight point -- not the same as IHIT for ASMs
	LASTPAY	Sortie position of last paying sortie point, preditch
	LASTTGT	Sortie position (=NUMHIT + 1) for last target
	NOMIT	Number of targets in potential target array not currently in sortie
	IOMIT(25)	SORTYTGT index of Ith omitted target
	NAVAIL	Number of s[illegible]ces in potential target arrays available for new targets
	IAVAIL(25)	SORTYTGT index of Ith available space
	LKHITMT(25)	Link for target J to sortie hit list if positive; omit list if negative
	ITOLD	Pointer used in finding lost targets

3.6.12 Subroutine GETGROUP

PURPOSE:	To read the BASFILE, one weapon group at a time, and call PRERAID or MISAGN to process the bomber or missile group.
ENTRY POINTS:	GETGROUP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ARAYSIZE, ASMTABLE, BEGIN, C4, C7, C9, CONTROL, DEBUG, DPENREF, FILABEL, FILES, GRPTYPE, IDUMP, IFTNO IFTPRNT, INPUTFL, ISKIPTO, ITP, MASTER, MYIDENT, MYLABEL, NOPRINT, PAYDATA, PAYLOAD, PCALL, PLANTYPE, POSTHL, PRINTOPT, STRKSUM, TWORD, VAL, WAROUT
SUBROUTINES CALLED:	ABORT, DEACTIV, MISASGN, OUTSRT, PRERAID, PRINTIT, PRNTF, RDARRAY, RDWORD, SETFLAG, SETREAD, SETWRITE, SKIP, TERMTAPE
CALLED BY:	POSTALOC

Method:

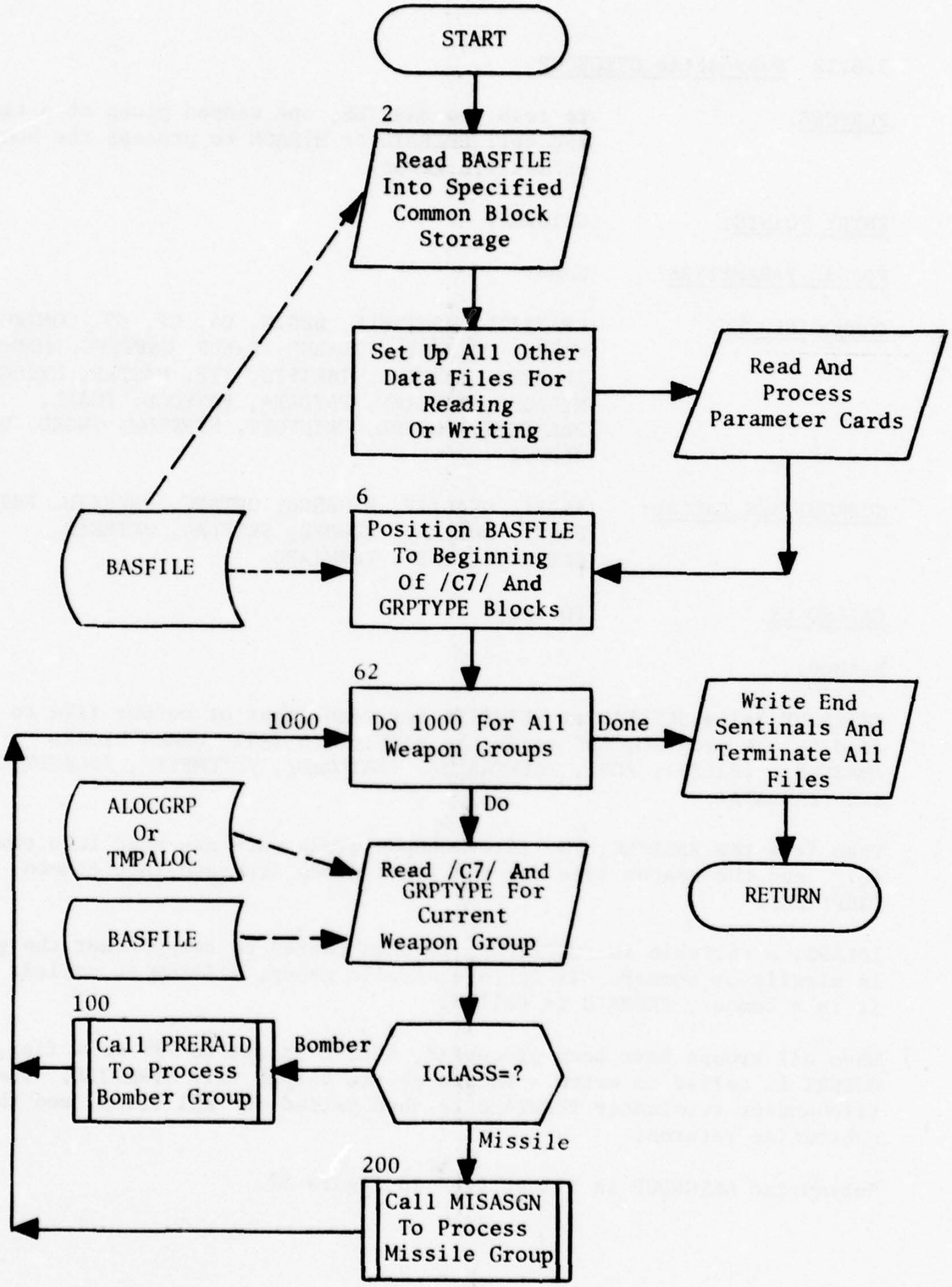
GETGROUP calls SETREAD or SETWRITE for each input or output file to be used by the program; and reads the BASFILE to fill common blocks /MASTER/, /FILES/, /C9/, /ASMTABLE/, /PAYLOAD/, /DPENREF/, /PLANTYPE/, and /PAYDATA/.

Then from the BASFILE, the first weapon group data are read into common /C7/, and the weapon type for the first group are read into common /GRPTYPE/.

ICLASS, a variable in /GRPTYPE/, is then tested to see whether the group is missile or bomber. If it is a missile group, MISASGN is called; if it is a bomber, PRERAID is called.

When all groups have been processed, IGROUP is set to 999 as a flag, and OUTSRT is called to write a record on the output file STRKFILE. The filehandler terminator TERMTAPE is then called for all files, and the subroutine returns.

Subroutine GETGROUP is illustrated in figure 57.



igure 57. Subroutine GETGROUP

3.6.19 Subroutine OPTRAID

PURPOSE: To control the setting up, optimizing, storing, and writing on tape of the sortie plans, for each sortie in the raid.

ENTRY POINTS: OPTRAID

FORMAL PARAMETERS: None

COMMON BLOCKS: C8, DEBUG, FLTPAS, INDEX, PCALL, POSTHL, PRINTOPT

SUBROUTINES CALLED: DUMPSRT, GETSORT, INITOPT, OUTSRT, PRINTIT, PRNTF, SETFLAG, SORTOPT, TIMEME

CALLED BY: GENRAID

Method:

OPTRAID controls the cycling of the optimization from one sortie to the next within a raid. It is called by GENRAID after FLTROUTE and TGTASGN have completed an initial assignment of strikes to sorties for all sorties in the raid (or subraid). For each sortie, OPTRAID makes calls on the appropriate data handling routines to set up the sortie for optimization, and it then calls SORTOPT to accomplish the optimization.

Before beginning to process the sorties in the raid, OPTRAID makes a call on INITOPT. The purpose of this call is to clear out the SORTYTGT arrays and initialize them for the new raid.

OPTRAID makes two optimization passes over the sorties, the second in reverse order. One reason for the double pass over the sorties is to provide a chance for valuable targets omitted by a sortie to be picked up by sorties on either side. On each pass, before calling SORTOPT, OPTRAID calls GETSORT. GETSORT reads the list of targets for the sortie as prepared by TGTASGN and inserts the appropriate targets into the SORTYTGT arrays. GETSORT also sets up the current definition of the sortie in the CURSORTY array.

SORTOPT is then called to optimize the sortie, and DUMPSRT is called to record the resulting sortie. Targets that remain in the sortie are cleared out of the SORTYTGT array and are recorded for the skeletal representation of the sortie in the JTGT array of common block /C5/. (The indices for ASM targets are recorded here with a minus sign.) Similarly, if the resulting optimized sortie does not include recovery, this is also noted in the skeletal representation (by making the number assigned NASGN negative in common /C6/. Targets that are omitted from the sortie by SORTOPT are not cleared out by DUMPSRT, so that they remain in the IOMIT list for future consideration.

On the second pass, subroutine OUTSRT is called after SORTOPT. OUTSRT records the final form of the sortie, including specific distances at low altitude, on the output STRKFILE to be used by PLANOUT.

Subroutine OPTRAID is illustrated in figure 64.

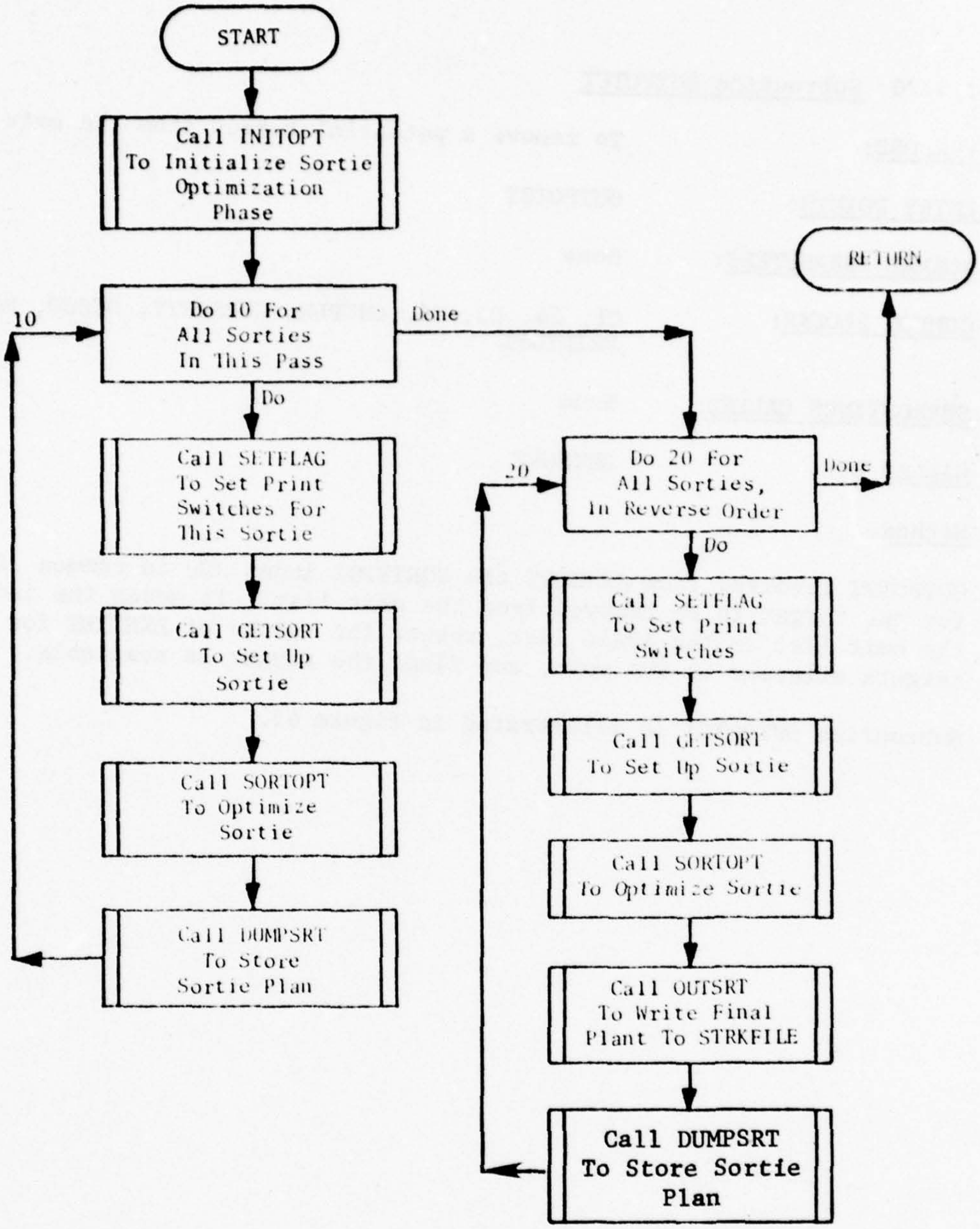


Figure 64. Subroutine OPTRAID

3.5.20 Subroutine OUTPOTGT

PURPOSE: To remove a potential target from the omit list.

ENTRY POINTS: OUTPOTGT

FORMAL PARAMETERS: None

COMMON BLOCKS: C1, C4, C5, C6, CHGPLN, CURSORTY, DEBUG, POSTHL, PRINTOPT

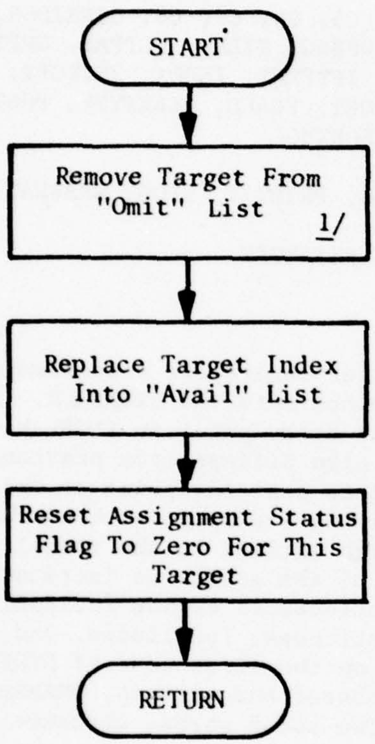
SUBROUTINES CALLED: None

CALLED BY: GETSORT

Method:

OUTPOTGT receives from GETSORT the SORTYTGT index JDO in common /CHGPLN/ for the target to be removed from the omit list. It moves the index from the omit list to the AVAIL list, resets the values of LKHITMT for the targets affected by the move, and flags the target as available.

Subroutine OUTPOTGT is illustrated in figure 65.



Note:
1. **IOM** is set to the position of the target JX in the omit list (=-LKHITMT(JX)), where JX has been set equal to JDO.

Figure 65. Subroutine OUTPOTGT

.5.21 Subroutine OUTSRT

PURPOSE: To write the final bomber sortie plan onto the output STRKFILE for input to PLNTPLAN.

ENTRY POINTS: OUTSRT

FORMAL PARAMETERS: None

COMMON BLOCKS: BEGIN, C1, C3, C4, C5, C6, C7, C8, CORRIDOR, CURSORTY, DRECOV, DEBUG, FILES, FLTPAS, GRPTYPE, HOB, IDUMP, IFTNO, IFTPRNT, INDEX, INITOPT, INPUTFL, ITP, OUTSORT, PCALL, PLANTYPE, POSTHL, PRINTOPT, PRINTF, SORTNO

SUBROUTINES CALLED: ABORT, FINFLT, GLOG, PRINTIT, SLOG, WRARRAY

CALLED BY: GETGROUP, OPTRAID, REFABORT

Method:

All of the data to be included in the output record are assembled in common /OUTSORT/ to be written out as a block onto the STRKFILE. A call is made on FINFLT, which is a special entry point in FLTPLAN, to produce a final sortie plan. This final plan differs from previous plans in that, if there are defended targets past the point at which low altitude ran out, FINFLT considers flying low up to each of these defended targets (and therefore flying high earlier in the route), checking at each leg to see if the value of the sortie is increased. When the final plan is obtained, it is included in common /OUTSORT/ as a series of happenings with associated latitudes, longitudes, and object numbers. User's input TARFAC is written on the first word of STRKFILE. The Sortie Sequence Number, ISORTN, is updated and written, WRARRAY is then called to write common /OUTSORT/. Two added words, distance to recovery and origin, are written.

During processing the input HOB information is transferred from common /HOB/ to array MYHOB in common /OUTSORT/.

To guarantee all bomber sorties fully utilize their onboard weapon inventory, strikes are artificially assigned where necessary. This is conducted between labels 71 to 10,000. For an under-utilization of one strike, the first target is assigned to be hit twice. For conditions of more than one weapon not assigned, the first target is doubly assigned as well as the second and so on as necessary. If necessary, conditions may exist where the first (and subsequent targets) may require multiple assignments artificially generated.

Subroutine OUTSRT is illustrated, with annotation, in figure 66.

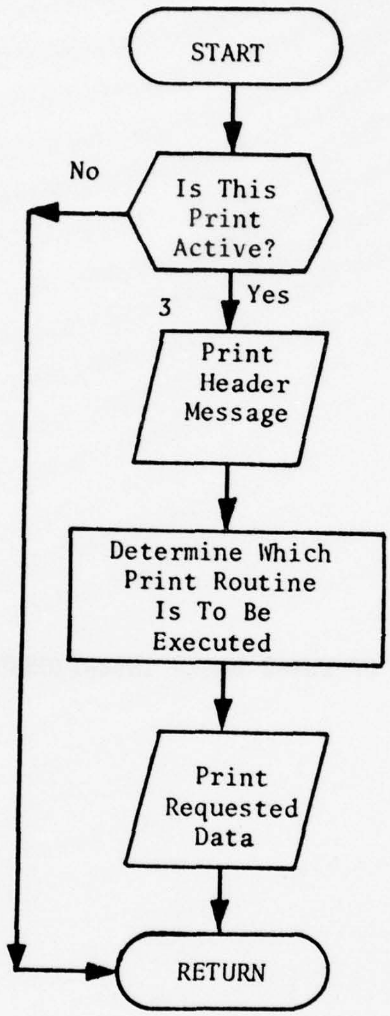


Figure 70. Function PRNTF

CONTENTS OF THESE PAGES INTENTIONALLY DELETED

CONTENTS OF THIS PAGE INTENTIONALLY DELETED

?.27 Subroutine SETFLAG

PURPOSE:	To read print request control cards from standard input data, and set print switches as requested.
ENTRY POINTS:	SETFLAG
FORMAL PARAMETERS:	None
COMMON BLOCKS:	C7, DATA, FLAG, INDEX, PCALL, WAROUT
SUBROUTINES CALLED:	SLOG
CALLED BY:	POSTALOC, GENRAID, OPTRAID, GETGROUP

Method:

The first time it is called, SETFLAG reads all the data cards into common /DATA/. A card with 99999 in the first field, or the 60th card, will terminate the read. The data cards consist of six format fields (I10) as follows:

Field	Contents
1	Value of ICALL at desired print request (print request number)
2	First sortie for which print is desired
3	Last sortie for which print is desired
4	OPTRAID pass (1 or 2) for which print is desired
5	Penetration corridor in which print is desired
6	Weapon group in which print is desired

A zero in any field implies no restriction at that level; i.e., a card with 36, 0, 0, 1, 3, 0 will turn the print at ICALL = 36 on for all sorties on the first pass of the third corridor in all groups.

SETFLAG is called by GENRAID whenever a new corridor is processed, and is called by OPTRAID whenever a new sortie is processed. Thus on each call, it tests the current sortie, pass, corridor, and group against each print request in common /DATA/ and, if the conditions are right, it activates the print by setting to 1 the ICALLth cell of the IFLAG array in common /FLAG/.

Subroutine SETFLAG is illustrated in figure 72.

3.6.29 Subroutine TGTASGN

PURPOSE: To make the initial assignment of targets to all sorties in a flight.

ENTRY POINTS: TGTASGN

FORMAL PARAMETERS: None

COMMON BLOCKS: ARAYSIZE, C2, C3, C4, C5, C6, C8, DEBUG, FIXALL, INDEX, IRESRCH, PCALL, POSTHL, PRINTOPT, TGTASIN,

SUBROUTINES CALLED: PRINTIT, PRNTF, TIMEME, GLOG

CALLED BY: FLTROUTE

Method:

TGTASGN is called by FLTROUTE to carry out the actual assignment of strikes to each sortie in a flight. The first and last sortie for the flight are specified by FLTROUTE (IFSTVEH and LSTVEH in common /TGTASIN/) together with a flag(ISIDE) which notes whether targets are to be assigned from the left side of the corridor toward the middle or from the right side of the corridor toward the middle. For each side of the corridor, TGTASGN maintains a pointer to the first unassigned target so that the scanning of targets to be assigned can begin with this target.

Figure 74 illustrates the operation of TGTASGN. For each new sortie the "target limit" (i.e., the total number of targets that should be assigned up to and including the sortie) is increased by the number of warheads on the sortie (NWPV) multiplied by the average number of strikes per warhead for the corridor (TGTSPWHD). Since the allocator has assigned more strikes than it has weapons, the number of strikes per warhead is usually a little more than 1.0. Therefore (because no fractional strikes can be assigned) the number of strikes per sortie will vary and an occasional sortie will be assigned an extra strike in excess of the available warheads.

Specific strikes are then selected for the sortie, beginning with the first unassigned strike from either the top or bottom of the list depending on the value of ISIDE, until the target limit is exceeded.

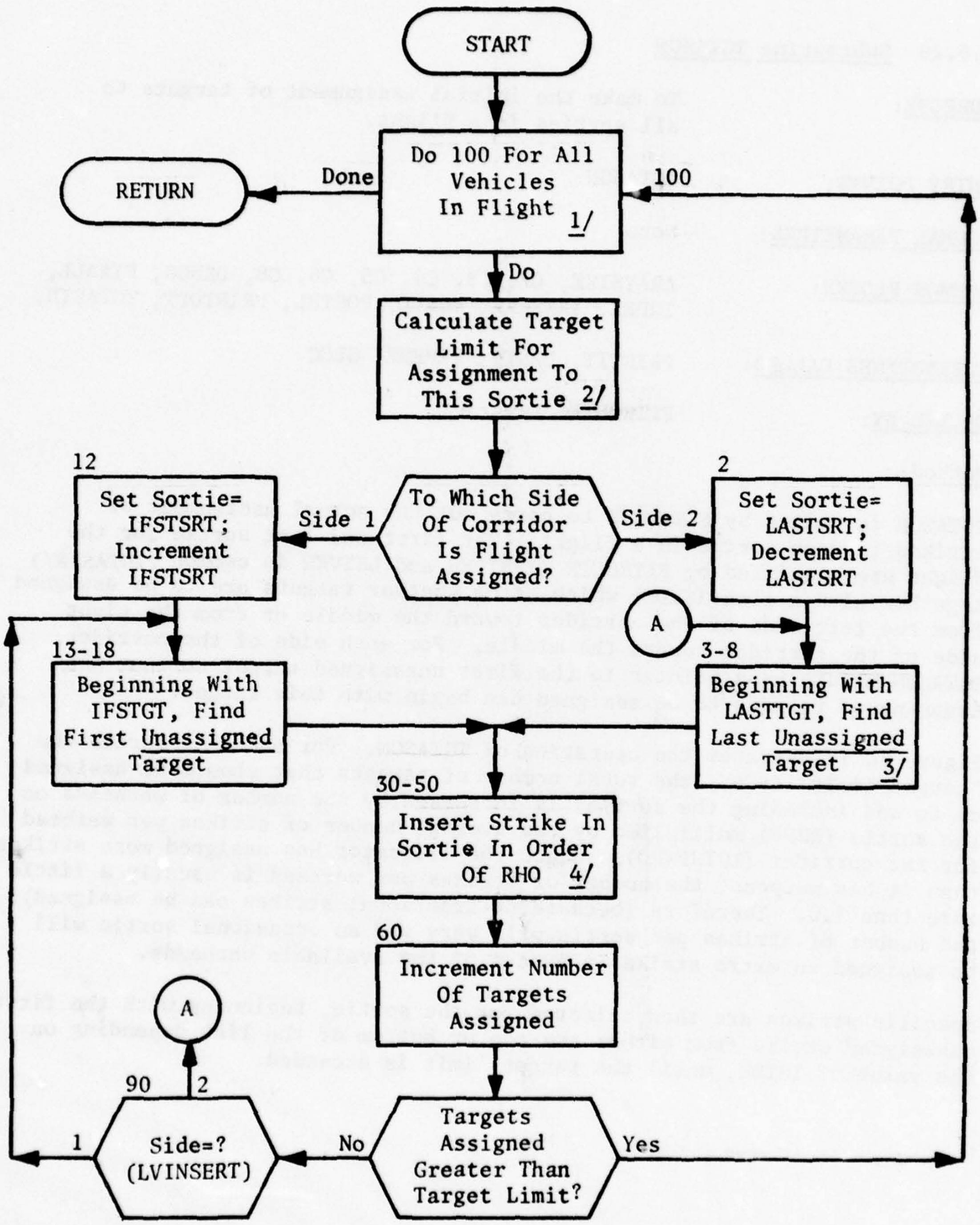


Figure 74. Subroutine TGTASGN (Part 1 of 2)

Notes:

1. TGTASGN receives from FLTROUTE the variables IFSTVEH and LSTVEH, indicating which sorties are to be processed. It initializes variables IFSTSRT and ILASTSRT to those values and uses then as internal markers.

2. TGTLIM, the target limit, is calculated as a floating point number. Therefore, since TGTSPWHD is usually slightly greater than 1.0, occasionally a sortie will be assigned more targets than it has warheads. These extra targets will be dropped in the sortie optimization and will be considered for substitution in other sorties.

3. When TGTASGN is entered, IFSTGT and ILASTGT are set at the first and last unassigned target, respectively. MYASGN is initially set to -1 for all targets. As soon as TGTASGN "looks" at a target, it sets MYASGN to 0. If it assigns the target to a sortie, it sets MYASGN to 1. Thus the first unassigned target may be 0, if the target has been considered previously and rejected, or -1, if it is being looked at for the first time.

4. The list of targets for a given sortie are listed in order of RHO as an initial attempt at determining the flight route of the sortie. This order is subject to change by EVALB which does a sequence check during its evaluation.

Figure 74. (Part 2 of 2)

THIS PAGE INTENTIONALLY LEFT BLANK

Section Page

ction		Page
5	Deleted	645

ILLUSTRATIONS (PART II)

Figure | Page

TABLES (PART II)

ABSTRACT

The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, provide output summaries, and provide input tapes to simulator subsystems external to QUICK. QUICK has been programmed in FORTRAN for use on the CCTC HIS 6000 computer system.

The QUICK Program Maintenance Manual consists of four volumes: Volume I, Data Management Subsystem; Volume II, Weapon/Target Identification Subsystem; Volume III, Weapon Allocation Subsystem; Volume IV, Sortie Generation Subsystem. The Program Maintenance Manual complements the other QUICK Computer System Manuals to facilitate maintenance of the war gaming system. This volume, Volume IV, is in two parts providing the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the programs and subroutines of the Sortie Generation subsystem. The associated program listings which are dynamic and voluminous are not contained herein. However, the program listings may be obtained by arrangement with the CCTC QUICK Project Officer. Companion documents are:

a. USERS MANUAL
   Computer System Manual CSM UM 9-77, Volume I
   Computer System Manual CSM UM 9-77, Volume II
   Computer System Manual CSM UM 9-74, Volume III
   Computer System Manual CSM UM 9-74, Volume IV
   Provides detailed instructions for applications of the system.

b. TECHNICAL MEMORANDUM
   Technical Memorandum TM 153-77
   Provides a nontechnical description of the system for senior management personnel.

SECTION 4. PROGRAM PLANOUT

4.1 Purpose

PLANOUT's sole function is the execution of three separate overlays: PLANO1, PLNTPLAN and INTRFACE.

The first overlay, PLANO1, permits the user to make minor changes to the plans generated by QUICK without requiring a new allocation. PLANO1 gives the user control over targets assigned to a specific sortie, the weapon offsets time of delivery, and height of burst. The user may change plans generated by POSTALOC or a recycling of PLANOUT.

PLNTPLAN, the second overlay, accepts output from PLANO1 (which includes the user target changes) and finalizes the sorties generated by the previous programs in plan generation.

The last overlay, INTRFACE, adds information to the output of PLNTPLAN, and creates tapes to be used in programs external to QUICK.

4.2 Concept of Operation

PLANO1 processes sortie change cards as input by sortie sequence number, change operation codes, and target designator codes. PLANO1 merges these changes with the STRKFILE or, if desired, from a previous PLANOUT output file. The output Strike Change (STRKCHNG) file from PLANO1, along with STRKFILE will be used in PLNTPLAN (second overlay) to finalize plans.

The sortie change cards permit the user to make minor target changes on any desired sortie but not requiring a new allocation. Via user inputs, controls may be exercised over:

- The target designator (DESIG), input time, weapon height of burst, Desired Ground Zero (DGZ) coordinates, and depenetration corridor
- The capability to add, delete or replace complete strikes by sortie sequence number without having to specify times and so forth.

PLNTPLAN, the second overlay, accepts output from PLANO1 (which includes the user target changes) and finalizes the sorties generated by the previous programs in the Sortie Generation subsystem.

PLNTPLAN processes the bomber and missile plans given it on the STRKFILE by program POSTALOC or STRKCHNG from PLANO1, and a detailed plan is output via the PLANTAPE which reflects the plan in a form more suitable for

hard copy output. The PLANTAPE is used as input to INTRFACE and program EVALALOC. Detailed prints of the final plans may also be obtained from PLNTPLAN. Each type of plan (bomber, missile, and tanker) is handled differently. Among the processing functions performed on the input bomber plans are: assigning refuel areas; calculating ASM launch points; determining where altitude changes, and decoy launch points should occur; and coordinating launch times according to user parameters.

For missiles, launch times are assigned based on user-supplied coordination parameters. Tanker plans are generated such that all bombers will be serviced as required. Finally, PLNTPLAN calculates the distances and times between all events of each plan.

The last overlay, INTRFACE, adds information to the output of PLNTPLAN, and creates tapes to be used in programs external to QUICK.

INTRFACE is one of the two special-purpose processors which provide an interface between QUICK and other computerized simulation systems used in strategic war gaming. INTRFACE and the TABLE tape extract and reformat data from QUICK-developed files and output data which are used as input to the Event Sequenced Program (ESP), the Nuclear Exchange Model (NEMO), and to a CCTC damage assessment system, SIDAC (Single Integrated Damage Analysis Capability System). INTRFACE does not produce output files which are used within QUICK. Specifically, INTRFACE processes data contained on the PLANTAPE and reformats it for output on two tapes, the STRIKE tape (PTAPE), and the sortie specifications tape (ABTAPE).

4.3 Input Files

4.3.1 Input to Overlay PLANO1. Three or four files are input to PLANO1: BASFILE, STRKFILE, SCHNG1, and TARFILE. SCHNG1 refers to PLANO1 input file; SCHNG2 refers to PLANO1 output file (or PLNTPLAN input). A general reference to STRKCHNG (i.e. when discussing formats) refers to SCHNG1 or SCHNG2.

The STRKFILE is the output of POSTALOC, and contains the skeletal plans for each bomber and missile. Common /OUTSRT/ contains the input bomber; common /BLOCK/ contains the input record for missiles. The end-of-file signal is a dummy bomber record with a group number of 201. The first word on the file contains parameter TARFAC. This is an input to POSTALOC and its same value is required for any sortie add option in PLANO1.

A SCHNG1 file may or may not exist and if it does, it may not be used (via input option). This file is an output from a previous PLANOUT run and contains indicators for any sortie changed from the STRKFILE. If, for a given sortie, the indicator is positive a record similar to STRKFILE exists and is read into common /OUTSRT/ or /BLOCK/ thus

destroying the STRKFILE record. Additional change information is read into common /OUTSRA/. By using SCHNG1, changes may be made upon changes without requiring the input cards that generated any of the past updates. Also, SCHNG1 contains all those missile sorties that may have been added and do not exist on STRKFILE. An end-of-file is indicated by a sortie sequence number greater than 99999.

The TARFILE, created in program PLANSET, contains LAT, LONG, INDEXNO, TASK, CNTRYLOC, TARDFH1, and FLAG for each target designator code. These parameters are required if a target is added or replaces an old one. This file is accessed by hash code on the DESIG of the new target. Header information is read into common /TARLIST/.

BASFILE, created by program PREPALOC, provides necessary information if missiles are to be added as a new sortie and bomber corridor parameters are read for low altitudes and attrition recalculation. Data is read into common /BASEF/, /CORRCHAR/, and /DPENRE/.

4.3.2 Input to Overlay PLNTPLAN. Four files are input to PLNTPLAN: the STRKFILE, STRKCHNG, BASFILE, and MSLTIME file.

The STRKFILE is the output of program POSTALOC, and contains the skeletal plans for each bomber and missile. Common block /OUTSRT/ contains the input bomber record; common /BLOCK/ contains the input record for missiles. The end-of-file signal is a dummy bomber record with a group number of 999. POSTALOC user input TARFAC is the first word on STRKFILE.

The input event list in common /OUTSRT/ includes all targets, whether bomb or ASM. It always begins with the corridor origin route leg, and ends with the input event DEPEN, LAND, or DIVEMISL (if an air-breathing missile), depending on whether the mission is successful or aborts. Table 16 lists the admissible input events by type, and indicates what information in the list is relevant for each type.

Table 16. List of Admissible Input Events by Type and Information Relevant to Each

TYPE OF HAPPENING	LAT., LONG	PLACE	WEAPON OFFSET LAT., LONG.
DOGLEG	Y	N	N
DROPBOMB	Y	Target Index	Y
AIM ASM	Y	Target Index	Y
DEPEN	N	Depenetration Corridor Index	N
LAND	N	N	N
DIVEMISL	N	N	N

Y = relevant

N = not relevant

Bomber sorties are identified uniquely by sortie number, group index, and corridor index together. Missiles are identified by index number.

PLANO1 creates the STRKCHNG by merging the STRKFILE with target changes supplied by the user. The file consists of a header record followed by a short block for each sortie containing the sortie sequence number and an indicator to show whether the STRKFILE has been changed. If a change exists, the altered plan follows; and common /OUTSRT/ or /BLOCK/ is redefined. Additional change plan information is read into common /OUTSRA/.

STRKCHNG contains indicators for all sorties on the STRKFILE plus missile sorties completely added by the user. The end-of-file signal is a sortie sequence number greater than 99999.

The BASFILE, created by program PREPALOC, provides such information as corridor, payload, weapon, and tanker data. It is from this file that PLNTPLAN retrieves the data required to fill the following common blocks:

/MASTER/

/FILES/

/CORRCHAR/

/PAYDATA/

/ASMTABLE/

/PAYLOAD/

/DPENREF/

/PLANTYPE/

/WPNTYPEX/

/WPNGRPX/

/NAVAL/

/C9/ (Variables corresponding to /CHARTER/, and /HAPPEN/ in other QUICK programs)

/C7/

The MSLTIME file, created by program ALOC, contains a five-word record for each fixed missile (i.e., each missile-delivered weapon). It is read into array MSLFIL of common block TIMELINE. If there are no fixed missile assignments in the current plan, it consists simply of an end-of-file record.

4.3.3 Input to Overlay INTRFACE. The QUICK developed files used by INTRFACE are the BASFILE prepared by program PREPALOC and the PLANTAPE output by overlay PLNTPLAN. The BASFILE is used to obtain weapon and vehicle type data which are not included on the PLANTAPE. The PLANTAPE provides detailed information about missile, bomber, and tanker plans prepared by the QUICK Plan Generator.

4.4 Output Files

4.4.1 Output Files from Overlay PLAN01. SCHNG2 is the only generated output. STRKFILE and SCHNG2 are used in PLNTPLAN to generate the finalized plans. By saving spill tapes at the completion of PLANOUT, SCHNG2 may be recycled for additional sortie target changes in subsequent runs.

SCHNG2 file consists of a header record (table 17) followed by a short block for each sortie containing the sortie sequence number and an indicator to show whether the sortie has been changed. If the sortie has been changed, a record follows similar to a STRKFILE record for the type of sortie but with arrays that contain additional information concerning how a target has been changed. Tables 18 and 19 show the bomber and missile records on SCHNG2.

Table 17. STRKCHNG File Header Record

WORD	DESCRIPTION
1-30	Latitude of corridor point
31-60	Longitude of target point
61-90	Defense zone in which corridor origin is located
91-120	Latitude of corridor origin
121-210	Average attrition per distance in Jth corridor, Ith leg
211-240	Distance from corridor entry to corridor origin
241-270	Parameter to adjust mode of corridor penetration
271-300	High-altitude attrition per nautical mile unsuppressed
301-330	High-altitude attrition per nautical mile suppressed
331-360	Ratio low-to-high altitude attrition
361-390	Characteristic range of corridor defense (nautical miles)
391-420	Number of attrition sections this corridor
421-510	Distance in each precorridor leg
511-600	Attrition in each precorridor leg
601	Number of words in common /CORRCHAR/
602-631	Total precorridor defended distance, corridor J
632-721	Order of importance of attrition in Jth corridor, Ith leg

4.4.2 Output Files from Overlay PLNTPLAN. The principal output of PLNTPLAN is the PLANTAPE containing the bomber, missile, and tanker plans for use by overlay INTRFACE and program EVALALOC. The admissible events for the PLANTAPE record are those listed in table 20.

Table 20. Bomber Events Recognized by PLNTPLAN

TYPE OF EVENT	EVENT TYPE	PLACE INDEX	EVENT NAMES USED IN /EVENTS/	EVENT NAMES USED IN OUTPUT PRINT
Launch	2	Base Index	LAUNB	LAUNCH B
Refuel	4	Refuel Index	LEREFUEL	REFUEL
Local Attrition or Drop Bomb	8	Target Index	LOCLATTR	DROPBOMB
Launch ASM	14	ASM Type	LAUNASM	LAUN ASM
ASM Target	--	Target Index	--	ASM TGT
Launch Decoy	15	*	LAUNDCOY	LAUNDCOY
Change Altitude	17	1 for Go High, 0 for Go Low	LOHI	CHANGALT
Recover	16	Recovery Base Index	LANDHO	RECOVER
Abort	13	*	LABORT	ABORT
Enter Refuel	11	Refuel Area Index	LENTEREF	ENTERREF
Leave Refuel	12	*	LEAVEREF	LEAVEREF
Go High	18	*	IGOHI	GO HIGH
Go Low	19	*	IGOLOW	GO LOW
Dogleg	20	*	LEGDOG	DOGLEG

* Place index not applicable

CONTENTS OF THESE PAGES INTENTIONALLY DELETED

PLANTAPE is retrieved from the arrays of common blocks /INDATA/, /DINDATA/, and /DINDT2/ for bombers and tankers. For missiles, it is written from variables within common /BLOCK/. The tape format for these records is shown in tables 26 through 29. The end of this first file (containing missile, bomber and tanker plans) is sign signaled by a dummy record containing 25 words of zeros, followed by an END FILE mark.

The second file on the PLANTAPE contains the refuel area table. The first word is the number of refuel areas (NRF). Then follow NRF word pairs containing the latitude and longitude of each refuel area.

The PLANTAPE is written using standard FORTRAN unformatted WRITE statements rather than the filehandler subroutines.

This record precedes the Launch event for the first consecutively launched vehicle from a base of weapons having a time-dependent destruction before launch probability.

CONTENTS OF THIS PAGE INTENTIONALLY DELETED

Table 26. Format of PLANTAPE Record (Bomber Plans)
(Part 1 of 2)

Header Block:

WORD	DESCRIPTION
1	Sortie Sequence Number
2	Side
3	Group number
4	Penetration corridor number
5	Bomber sortie number
6	Base index number
7	Vehicle index number
8	ICLASS = 2
9	Weapon type
10	Launch region
11	Alert status
12	Payload index
13	Depenetration corridor number
14	Total number of bomber events in table
15	Number of planned bomber events (i.e., excluding Refuel/Abort missions events)
16	Available low-altitude range in precorridor legs
17	Available low-altitude range before first target
18	Available low-altitude range after first target
19-20	Lower plot markers for sortie
21-22	Upper plot markers for sortie
23	Weapon type name (ISIMTYPE)
24	Bomber type (Plan Generator type)
25	Bomber function code
26	Number of targets in total plan (set to 4HLAST after last good record as an end sentinel)

Table 31. (Part 2 of 3)

CARD COLUMN	VARIABLE NAME	INFORMATION	CONTENT
15-19	IDES	Location identifier for given operation code. The contents column shows the entry associated with the following codes.	
		IOP= 1	Base index INDEXNO
		= 2	Area number
		= 3	Zeros
		= 4	Zeros
		= 6	00001
		= 7	Target DESIG Code
		= 8	00001
		= 9	00001
		=10	Target DESIG Code
		=11	Target DESIG Code
		=13	Recovery base Name if bomber
20-25	LAT	Latitude at end of leg	(Degrees, minutes, seconds)
26-33	LON	Longitude at end of leg	(Degrees, minutes, seconds)
34	MODE	Mode of operation	1 - High altitude 4 - Low altitude
35		Zero	0
36-41	ICUMTIME	Time of event	Hours, minutes, and seconds
42	ISOUTH	Southern Latitude indicator	S if southern latitude, blank if not

Table 31. (Part 3 of 3)

CARD COLUMN	VARIABLE NAME	INFORMATION	CONTENT
43-44	ISCP	Sequential warhead number	01-10
45		Zero	0
46		Blank	
47-49	CAZIM	Launch/Back Azimuths in degrees	000 - 360
50	IECM	ECM	0 - Off 1 - On
51		Zero	0
52-53	IPAR1	Warhead type	01-99
54	IHXX	Height of burst (HOB)	0 - Ground
55-66		Zeros	1 - Air
57-58	IPLNETYP	Plane type code	01-99
59-60	ICNTRY	Code for country of target location	2 Alpha
61	IREGB	Region code	1-9
62		Blank	
63-64	ITASK	Target task code	2 Alpha
65-67	IHXXX	Height of burst (hundreds of feet)	000-999
68-72	LYIELD	YIELD(MT)	00001-99999
73-75	JCEP	000-999	CEP (100's feet)
76-77	ICOWN	Code for country of target owner	2 Alpha

Table 32. PLAN01, External Common Blocks (Part 1 of 5)

INPUT DATA FROM BASFILE

BLOCK*	VARIABLE OR ARRAY**	DESCRIPTION
/BASEF/		BASFILE parameter
	MSIDE	SIDE(BLUE=1; RED=2)
	TOFMIN(100)	Minimum time of flight for missile type (hours)
	CMISS(100)	Time of flight factor for missile type
	RNGMIN(100)	Minimum range for missile type. Used in calculating time-of-flight (nautical miles)
	IREG(250)	Region for weapon group
	ITYPE(250)	Weapon type for weapon group
	RANGEASM(40)	ASM range
	IASM(20)	ASM index
/CORRCHAR/		Corridor characteristics.
	PCLAT(30)	Latitude of corridor point
	PCLONG(30)	Longitude of corridor point
	RPLAT(30)	Latitude of corridor origin
	RPLONG(30)/ATPDST(30,1)***	Longitude of corridor origin
	ENTLAT(30)/ATPDST(30,2)***	Latitude of corridor origin
	ENTLONG(30)/ATPDST(30,3)***	Longitude of corridor entry
	CRLENGTH(30)	Distance from corridor entry to corridor origin
	KORSTYLE(30)	Parameter to adjust mode of corridor penetration
	ATTRCORR(30)	High-altitude attrition per nautical mile unsuppressed
	ATTRSUPP(30)	High-altitude attrition per nautical mile suppressed
	HILOATTR(30)	Ratio low- to high-altitude attrition (less than 1)

* Ordered alphabetically, not by position in core.

** Parenthetical values indicate array dimensions. All other elements are single-word variables.

***If STRKCHNG is read, arrays RPLONG, ENTLAT, ENTLONG are replaced by ATPDST(30,3); the average attrition per distance in Jth corridor, Ith leg.

Table 32. (Part 2 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/CORRCHAR/ (cont.)	HILOATTR(30)	Ratio low- to high-altitude attrition (less than 1)
	DEFRANGE(30)	Characteristic range of corridor defense (nautical miles)
	NPRCRDEF(30)	Number of attrition sections this corridor
	DEFDIST(30,3)	Distance of each precorridor leg
	ATTRPRE(30,3)	Number of words in common /CORRCHAR/
/DPENREF/	DPLAT(50)	Depenetration latitude
	DPLONG(50)	Depenetration longitude*
	RCBLAT(50,4)	Recovery base latitude*
	RCBLON(50,4)	Recovery base longitude*
	INDCAP(50,4)	Recovery base capacity*
/FILES/		Logical unit number and maximum length for all Plan Generator files.
	TGTFILE(2)**	Target data file
	BASFILE(2)	Data base information file
	MSLTIME(2)	Fixed missile timing file
	ALOCTAR(2)	Weapon allocation by targets file
	TMPALOC(2)	Temporary allocation file
	ALOCGRP(2)	Allocation by group file
	STRKFILE(2)	Strike file
	PLANTAPE***	Detailed plans tape

* Indexed for each of four bases assigned to each depenetration point.
** In two-word arrays, first word is logical unit number; second word is maximum file length in words. Single variables are logical unit numbers.
***These files are output on magnetic tape.

Table 32. (Part 3 of 5)

INPUT DATA FROM CARDS

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/CARDS/		Sortie change card parameters
	ICODE	Operation code (A=add missile, C=change, I=insert, E=end)
	IHOB	Height of burst (A=air, G=ground
	ISTYP	Sortie type (blank=missile, B=bomber)
	IDCC	Depenetration corridor change (blank=no change; otherwise corridor number)
	IASM	Bomber weapon launch type (blank=bomb, A=ASM)
	IRAC	Recalculate attrition for bomber (blank=yes, N=no)
	IAS	Not used
	ISSN	Sequential Sortie Number
	IDESIG1	Target designator reference
	IDESIG2	Designator for new target (blank for delete)
	TCHANGE	Time change (minutes)
	CALOFF	If blank, desired offsets are defined by parameters DLATOF and DLONGOF. If equals to 'C', parameters DLATOF and DLONGOF define the actual ground zero. For this case, the offsets are calculated internally.
	DLATOF	Desired ground zero latitude offset (50ths of nautical mile) if parameter CALOFF is blank. Desired ground zero latitude (floating degrees) if paramter CALOF equals 'C'. If 'ZERO', offsets are set to numeric zero regardless of CALOFF entry.
	DLONGOF	Longitude offset. Identical definition as given for latitude.
	TLUCH	Time of launch (minutes)

Table 32. (Part 4 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/CARDS/ (cont.)	DLATOF	Desired ground zero latitude offset (50ths of nautical mile)
	DLONGOF	Longitude offset
	TLUCH	Time of launch (minutes)
	ILR	Launch region
	IALS	Alert status
	IMTI	Missile type index (plane type)
	IPI	Payload index
	IGI	Group index
	ISI	Site index
	LATD	Weapon site latitude (degrees)
	LATM	Weapon site latitude (minutes)
	LATS	Weapon site latitude (seconds)
	LATR	Weapon site latitude reference (N=North, S=South)
	LONGD	Weapon site longitude (degrees)
	LONGM	Weapon site longitude (minutes)
	LONGS	Weapon site longitude (seconds)
	LONGR	Weapon site longitude reference (E=East, W=West)
	ISAL	Salvo number
	JCARD	Card number (stored internally)
	NOLD	Save for last sortie number

INPUT (OR OUTPUT) FROM STRKCHNG FILE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/CORRCHAR/		STRKCHNG header (see /CORRCHAR/ input from BASFILE)
/CORRC1/		STRKCHNG header; plans follow
	TDEFDST(30)	Total precorridor defended distance
	IMPRTD(30,3)	Order of importance of attrition per nautical mile
/CONTR1/		File control parameter (fully described under Internal Common Block Section)
	LREAD(2)	Sortie Sequence Number and STRKCHNG indicator. Bomber or missile plans follow if indicator is nonzero; else STRKFILE plans are used for this sortie

Table 32. (Part 5 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/BLOCK/	LOCK/BLOCK(320)	Missile plan; identical to STRKFILE record (see STRKFILE format)
/OUTSRA/		Change arrays for missile plans
	HOBM(18)	Height of burst information
	DLTAM(18)	Change in time information
	INDRM(18)	Change indicator for targets
	IHOBM	Height of burst flag
	ICTIMEM	Change time flag
/OUTSRT/	---	Bomber plan; identical to STRKFILE record (see STRKFILE format)
/OUTSRA/		Change arrays for bomber plans
	DISTE(10)	Effective distance between targets
	HOB(10)	Height of burst information
	DLTA(10)	Change in time information
	INDR(10)	Change indicators for targets
	IOHOB	Height of burst flag
	ICTIME	Change time flag

INPUT DATA FROM STRKFILE FILE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/BEGIN/	TARFAC	TARFAC used in POSTALOC. Bomber and missile plans follow
/BLOCK/	LOCK/BLOCK(320)	Contains the input missile record from STRKFILE (see STRKFILE format)
/OUTSRT/	---	Contains the input bomber record from STRKFILE (see STRKFILE format)

INPUT DATA FROM TARFILE FILE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/TARLIST/		Target designator location indicator
	TARLIST(3516)	Storage array for given page of TARFILE
	SCAT(7771)	Scatter index array
	MAXPPG	Maximum number of pages
	MXSCAT	Length of scatter array

Table 33. Overlay PLNTPLAN External Common Blocks
(Part 1 of 6)

INPUT DATA FROM BASFILE

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
ASMTABLE	IWHDASM(20)	Warhead index
	RANGEASM(20)	Range
	RELASM(20)	Reliability
	CEPASM(20)	CEP
	SPEEDASM(20)	Speed
CORRCHAR	PCLAT(30)	Latitude of corridor point I
	PCLONG(30)	Longitude of corridor point I
	RPLAT(30)	Latitude of corridor I origin
	RPLONG(30)	Longitude of corridor I origin
	ENTLAT(30)	Latitude of corridor I entry
	ENTLONG(30)	Longitude of corridor I entry
	CRLENGTH(30)	Distance from corridor I entry to corridor I origin
	KORSTYLE(30)	Power of y versus x
	ATTRCORR(30)	High-altitude attrition per nautical mile unsuppressed
	ATTRSUPF(30)	High-altitude attrition per nautical mile suppressed
	HILOATTR(30)	Ratio low- to high-altitude attrition (less than 1)
	DEFRANGE(30)	Characteristic range of corridor defense
	NPRCRDEF(30)	Number of attrition sections this corridor
	DEFDIST(30,3)	Distance of attrition section
	ATTPRPE(30,3)	Attrition in this attrition section
	NDATA	Total number of words in common /CORRCHAR/
DELTA	LCHINTVL(75)	Launch interval according to missile type (hour)
	SIMLUNCH(75)	Number of simultaneous launches for salvoed missile types.
	ISAL(18)	Missile salvo number

*Parenthetical values indicate array dimensions. All other elements are single word availables.

Table 33. (Part 2 of 6)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DPENREF	DPLINK(50)	Depenetration point I link
	DPLAT(50)	Depenetration point I latitude
	DPLONG(50)	Depenetration point I longitude
	QFLAT(20)	Refuel point I latitude
	QFLONG(20)	Refuel point I longitude
FILES	TGTFILE(2)	Target data file (unit and maximum length)
	BASFILE(2)	Data base information file (unit and maximum length)
	MSLTIME(2)	Fixed missile timing file (unit and maximum length)
	ALOCTAR(2)	Weapon allocation by targets file (unit and maximum length)
	TMPALOC(2)	Temporary allocation file (unit and maximum length)
	ALOCGRP(2)	Allocation by group file (unit and maximum length)
	STRKFIL(2)	Strike file (unit and maximum length)
	PLANTAPE	Detailed plans tape
MASTER	IHDATE	Date of run which created BASFILE
	IDENTNO	Time of run which created BASFILE
	ISIDE	Side
	NRTPT	Number of route points
	NCORR	Number of corridors
	NDPEN	Number of depenetration corridors
	NRECOVER	Number of recovery bases
	NREF	Number of refuel areas
	NREG	Number of regions
	NTYPE	Number of weapon types
	NGROUP	Number of weapon groups
	NTOTBASE	Number of bases
	NPAYLOAD	Number of payload types
	NASMTYPE	Number of ASM types
	NWHDTYPE	Number of warhead types
	NTANKBAS	Number of tanker bases
	NCOMPLEX	Number of complex targets
	NCLASS	Number of weapon classes
	NALERT	Number of alert conditions
	NTGTS	Number of targets
	NCORTYPE	Number of corridor types
	NCNTRY	Number of country codes

Table 33. (Part 3 of 6)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
NAVAL	NNAVAL	Length of /NAVAL/ data arrays on BASFILE
	PKNAV(250)	Single shot kill probability against naval targets for group I
	ISVX(6)	Temporary storage array
	LSTDBLDS	Index of last base for which a DBL destruct event was written
PAYLOAD	NOBOMB1(40)	Number of bombs of type 1 (For MIRVs, the number of IRVs)
	IWHD1(40)	Type index of first bomb
	NOBOMB2(40)	Number of bombs of type 2
	IWHD2(40)	Type index of second bomb
	NASM(40)	Number of ASMs
	IASM(40)	ASM type
	NCM(40)	Number of countermeasures (bombers) Degradation factor (missiles)
	NDECOYS(40)	Number of decoys (for MIRVs, the number of terminal decoys per IRV)
	NADECOYS(40)	Number of area decoys
	IMIRV(40)	MIRV system identification number
PAYDATA	PAYALT(40)	Bomber weapon release altitude indicator
PLANTYPE	INITSTRK	Indicator for first or second strike
	CORMSL	Coordination time parameter for missiles
	CORBOMB	Coordination distance for bombers
WPNGRPX	ITYPEX (250)	Type number from BASEFILE
	DBLX (250)	DBL probability from BASFILE
WPNTYPEX	REL (100)	Weapon reliability from BASFILE
	IWHTYPE (100)	Type name from BASEFILE
	IFNCTN (100)	Function code for weapon
TANKER	INDEXTK	Tanker index
	TKLAT	Tanker latitude
	TKLONG	Tanker longitude
	IREFTK	Tanker refuel area index
	NPSQNTK	Number of tankers per squadron
	NALRTK	Number of alert tankers
	SPEEDTK	Tanker speed
	DLYALTK	Delay for alert tankers
	DLYNLTK	Delay for nonalert tankers

Table 33. (Part 4 of 6)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
TANKER (cont.)	TTOS	Total time on station
	ITYPETK	Tanker type
	TANKRNGE	Tanker range
C7	IINDEXTK(60)	Tanker base index
	TKRLAT(60)	Latitude of tanker base I
	TKRLONG(60)	Longitude of tanker base I
	IIREFTK(60)	Refuel area for tankers where n>0→ must refuel at area n
	NTKPSQN(60)	Number of tankers in squadron I
	NALRTNK(60)	Number of alert tankers at base I
	TANKSPD(60)	Speed of tankers at base I
	TKDLYALT(60)	Delay for alert tankers at base I
	TKDLYNL(60)	Delay for nonalert tankers at base I
	TKTTOS(60)	Total time on station
	IITYPTK(60)	Tanker type
	TRANGE(60)	Tanker range
C9*	ILAUNDEC(90)	Number of decoys launched
	TIMELAUN(90)	Time of decoy launch
	DISTORE(90,6)	Distance traveled by decoy
	HDTX(90)	Temporary line array
	KPLX(90)	Temporary place array
	JTPX(90)	Temporary event number array
	HLAX(90)	Temporary latitude array
	HLOX(90)	Temporary longitude array
	TZTX(90)	Temporary weapon offset latitude array
	TZNX(90)	Temporary weapon offset longitude
	IWHX(90)	Temporary weapon type index
	PAX(90)	Temporary probability of arrival array
	CMTX(90)	Temporary cumulative time array

***This common block is redefined when used in subroutines PLANTANK and VAM. The alternate definition is shown under Internal Common Block, Table 34.**

Table 33. (Part 5 of 6)

BLOCK

BLOCK	VARIABLE OR ARRAY	DESCRIPTION	
C9	KOUNT(30)		
Known elsewhere	IHAP(30)	Miscellaneous plan counters	
as /CHARTER/	MOUNT(50)	and indices	
	JHAP(50)		
	JAPTYPE(250)	Event type	
Known elsewhere	HAPLAT(250)	Event latitude	History
as /HAPPEN/	HAPLONG(250)	Event longitude	table
	HAPDIST(250)	Incremental distance	
	FILLR(666)	Not used	
C8	RECLAT(50)	Tanker recovery latitude	
	RECLONG(50)	Tanker recovery longitude	
C10	RCBLAT(50,4)	Recovery base latitude	Indexed
	RCBLON(50,4)	Recovery base longitude	for
	INDBAS(50,4)	Recovery base name	each of 4
	INDCAP(50,4)	Recovery base capacity	bases
	DISTR(50,4)	Distance to recovery	assigned
	TOF(50,4)	Time of flight to	to the
		recovery	Ith de-
			penetration
			point

INPUT DATA FROM STRKFILE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
BLOCK	LOCK/BLOCK(320)	Initially, contains the missile plan from STRKFILE (see STRKFILE format); also used to store the output plan record
OUTSRT	----	Contains the input bomber record from STRKFILE (see STRKFILE format)

Table 34. (Part 4 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
VEHIC	CEP1	Weapon CEP
	NVHC	The number of weapon vehicles
WAROUT	IWARFL	Logical unit number for war gaming print output

Table 35. PLAN01 Internal Common Blocks (Part 1 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/CONTROL/	LSIDE	Indicates which side
	NPLOT	Indicates number of plots per page
	ISZE	Plot size
	SCALE	Plot scale
/CONTR1/	ICHANGE	Less than zero, an error was found on cards; greater than zero, recalculate bomber attrition. ICHANGE is redefined in PLNTPLAN
	LREAD(2)	Sortie Sequence Number and Change indicator from STRKCHNG input
	MBOM	Number of words in common /OUTSRTA/ for STRKCHNG bomber plans
	MMIS	Number of words in common /OUTSRTA/ for STRKCHNG missile plans
	IST	STRKCHNG option: 0=do not use input STRKCHNG, ,=1, use input STRKCHNG
	LTST	Not used
	INDSK	Input logical unit number for STRKCHNG
	IOUTDSK	Output logical unit number for STRKCHNG
	INDSKM	Input logical unit name for STRKCHNG
	IOUTDSKM	Output logical unit name for STRKCHNG

*Parenthetical values indicate array dimensions. All other elements are single-word variables.

Table 35. (Part 4 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/XL/ (cont.) For Bomber Plans	JBTYPE(10)	Type of target
	XATTROT(10)	Local attrition
	JBJEC(10)	Target indices
	XDLAT(10)	Latitude of weapon offset
	XDLONG(10)	Longitude of weapon offset
	JBDES(10)	Designator code of target
	JTSK(10)	Task code of target
	JBCTY(10)	Country code of target
	JBFLG(10)	Flag of target
	XDSTE(10)	Effective distance
	JBHOB(10)	Height of burst
	XBDLTA(10)	Change in time
WAROUT	IWARFL	Logical unit number for war gaming print output

Table 36. Overlay PLNTPLAN Internal Common Blocks (Part 1 of 11)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
ARTIME	ARTIME(50)	Earliest bomber arrival time at refuel area I
	NBUDREF	Number of "buddy" refuelings required
	NBOMBREF(50)	Number of bombers assigned to refuel area I
	NTANKREF(50)	Number of tankers assigned to refuel area I
	IARVLS/ARVLS(2,1000)	ARVLS(1,I) = time of the Ith bomber refuel processed by PLNTPLAN ; IARVLS(2,I) = the refuel area for that bomber refuel
ASMARRAY	ALAT(10)	Aim point latitude
	ALON(10)	Aim point longitude
	IFLY(10)	Fly point flag
	IDIS(10)	Distance from fly point to ASM target
	IORD(10)	Sort index
	JAY	Index communicated to PREFL1, PREFL2
	DIST	Distance communicated to PREFL1, PREFL2, POSTFLY

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 36. (Part 2 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
CONTROL	LSIDE	Indicates which side
	NPLOT	Indicates number of plots per page
	ISZE	Plot size
	SCALE	Plot scale
/CONTR1/	ICHANGE	STRKCHNG read indicator for PLANTMIS. Zero implies no further read of missile plan; positive or negative read remaining plan; negative further implies plan is an added sortie on STRKCHNG. ICHANGE is redefined in PLANO1
	LREAD(2)	Sortie Sequence Number and change indicator from STRKCHNG input.
	MBOM	Number of words in common /OUTSRA/ for STRKCHNG bomber plans
	MMIS	Number of words in common /OUTSRA/ for STRKCHNG missile plans
	IST	STRKCHNG option (0=do not use input STRKCHNG, =1 use input STRKCHNG, =2 use input STRKCHNG in PLANO1 but not in PLNTPLAN)
	LTST	Not used
	INDSK	Input logical unit number for STRKCHNG
	IOUTDSK	Output logical unit number for STRKCHNG

Table 36. (Part 3 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/CONTR1/ (cont.)	INDSKM	Input logical unit name for STRKCHNG
	IOUTDSKM	Output logical unit name for STRKCHNG
CORCOUNT	IH	Points to line of /HAPPEN/ where current corridor begins
	KC	Number of lines in /HAPPEN/ describing current corridor
	JH	Points to line in /HAPPEN/ where current depenetration corridor begins (1 = depenetration point)
	LC	Number of lines describing current depenetration corridor (see common block /C9/ for description of /HAPPEN/)
DINDATA	HDT(90)	Time - for detailed history - of event I
	KPL(90)	Place - for detailed history - of event I
	JTP(90)	Event type - for detailed history - of event I
	HLA(90)	Latitude - for detailed history - of event I
	HLO(90)	Longitude - for detailed history - of event I
	TZT(90)	Weapon offset latitude - of event I
	TZN(90)	Weapon offset longitude - of event I
	PA(90)	Probability of arrival at target - of event I
	MHT	Total number of lines in detailed plan
	NPL	Number of planned events

Table 36. (Part 4 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DINDT2	CMT(90)	Cumulative time - for detailed history
	IWH(90)	Weapon type index - for detailed history
DISTC	DISTC(20)	Distances between target events
EVENTS	LAUNM	Launch missile code
	LAUNB	Bomber launch code
	LEREFUEL	Refuel code
	LOCLATTR	Local attrition or drop bomb event code
	LAUNASM	Launch ASM event code
	LAUNDCOY	Launch Decoy event code
	LANDHO	Recovery Event code
	LOHI	Change Altitude event code
	MISSATTR	Missile attrition event code
	LEGDOG	Dogleg event code
	LABORT	Abort event code
	LENTEREF	Enter refuel area event code
	LEAVEREF	Leave refuel area event code
	IGOHI	Go to high-altitude event code
	IGOLOW	Go to low-altitude event code
HILO	ISTOREHI	Number of event in /OUTSRT/ after which GO HIGH occurs

Table 36. (Part 5 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
HILO (cont.)	ISTORELO	Number of event in /OUTSRT/ after which GO LOW occurs
	IGOLEFT	Set to 1 if GO LOW range is available after depenetration
	FACHI	Distance after event ISTOREHI at which GO HIGH is located
	FACLO	Distance after event ISTORELO at which GO LOW is located
	GOLO	Amount of GO LOW range remaining for depenetration
ICLASS	IBOMBER	Bomber class index
	ITANKER	Tanker class index
IDP	IDP(2)	Depenetration corridor index number as reassigned when last target is an ASM target; IDP(1) is for primary plan, IDP(2) is for the alternate
IFLGDPEN	IFLGDPEN	= 1 at calculated GO LOW
		= 2 if GO LOW precedes first insector
IFLGDPEN	ICIFLG	Tactical aircraft flag
IGO	IGO800	Set to 1 for degenerate target area
IOUT	LPAYLOAD	Index of current payload
	LREF	Index of current refuel area

Table 36. (Part 6 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
IOUT (cont.)	LDPEN	Index of current depenetration point
	KOKO	Index of current ASM type
	JFCTNO	Function number of vehicle
IRF	IRF	Assigned refuel area index
	NRF	Number of refuel areas, including those assigned by PLNTPLAN
IRFTK	IRFTK	Refuel area index
ISRTNS	ISRTNS	Bomber sortie sequence number for print
KEYLENG	LOS	Length of /OUTSRT/ record (STRKFILE)
	LIN	Length of /INDATA/ record (EVENTAPE)
	LDN	Length of common block /DINDATA/
	LINC	Length of /INDATA/ except for last array
	LTK	Length of tanker record
	LMIS	Length of missile record, STRKFILE

Table 36. (Part 7 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
KEYLENG (cont.)	LMO	Number of good words in missile /INDATA/
	LDBL	Length of record for naval DBL event
LASM	U1	Latitude of beginning point of bomber path
	V1	Longitude of beginning point of bomber path
	U2	Latitude of end point of bomber path
	V2	Longitude of end point of bomber path
	UAT	Latitude of ASM target
	VAT	Longitude of ASM target
	RASM	Range of ASM
	RLAT	Latitude of ASM aim point
	RLONG	Longitude of ASM aim point
LAUNSNAP	INRANGE	Set to zero if ASM target is in range of flight path; otherwise to one
	FRACPATH	Fraction of total path at which ASM is launched
MH	MHMINA(10)	Line in common /DINDATA/ where target area begins
	MHMAXA(10)	Line in common /DINDATA/ where target area ends
	MHMN	Lower plot marker for sortie
	MHMX	Upper plot marker for sortie
MH2	MHMIN(2)	Lower plot markers for sortie
	MHMAX(2)	Upper plot markers for sortie

Table 36. (Part 8 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
MISCT	MISCT	Missile booster count
	MTARGCT	Missile target count
MRVFLG	MRVFLG	Set to 1 if plan contains MIRVs
POLITE	S1	Latitude of beginning interpolation point
	T1	Longitude of beginning interpolation point
	S2	Latitude of interpolation end point
	T2	Longitude of interpolation end point
	FACTOR	Interpolation factor or fraction
	SR	Latitude of interpolated point
	TR	Longitude of interpolated point
PPINFO	NDUMCORR	Tactical aircraft corridor index
	GOLOX1	Saved low altitude range available for use in corridor
	GOLOX2	Saved low altitude range available for use before first target
	GOLOX3	Saved low altitude range available for use after first target
	INDEX	Bomber plan index. Equals group number plus 100 times corridor index plus 10000 times sortie number
	NSORTIES	Total number of primary bomber plan processed
	INDXX	Group weapon type index

Table 36. (Part 9 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
PPINFO (cont.)	GOGO	Saved low altitude range available for use in corridor
	MHIST	Maximum number of entries into history table
	DUST	Distance bomber traveler during first history event
	LXIDPCHK	Logical array indicating if depenetration corridor is used
RECOVERY	NAMECAP(200,3)	J=1, Recovery base name J=2, Recovery base capacity J=3, Number of aircraft that landed at recovery base
	NBOMGP	Number of bomber groups
	NUSED(200)	Working array that defines the number of aircraft arriving at each base for a given group
	IREC	Logical unit containing recovery base data
	NREC	Number of unique recovery bases
REF	RFLAT(50)	Latitude of refuel area I
	RFLONG(50)	Longitude of refuel area I
RL	RL	Decoy low-altitude range
	RH	Decoy high-altitude range
SNAPON	NAP(15)	Set to three for active print I; set to one for inactive print I
SPASM	SPASM	Speed of ASM currently used
TEMPO	DT(50)	Distance or time temporary storage
	JT(50)	Event type temporary storage
	TLT(50)	Latitude temporary storage

Table 36. (Part 10 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
TEMPO (cont.)	TLN(50)	Longitude temporary storage
	LPL(50)	Place index temporary storage
TIMELINE	ITIMETYP(40)	CORMSL type (0-Flight; I-Line)
	CORMSLX(40)	Percent flight complete or time on line
	FLTMIN(40)	Minimum flight time in minutes
	INDXFIX(1000)	Target index numbers for fixed weapons
	TIME(1000)	Arrival times for fixed weapons
	TDATA(252)	Temporary storage area for flight data
	ZLAT(50,2)	Latitude of timing line end-points
	ZLONG(50,2)	Longitude of timing line endpoints
	XC(50)	X-coordinate of cross product vector of timing line
	YC(50)	Y-coordinate of cross product vector of timing line
	ZC(50)	Z-coordinate of cross product vector of timing line
	DL(50)	Length of timing line
	TMLNCH(18)	Launch time
	GOLD	Last fixed weapon group processed
	NFIXWPS	Number of fixed weapons
	NLEFT	Counter for fixed weapons
	NLINES	Number of timing lines
	MSLFIL(5)	Input area for record from MSLTIME file (see MSLTIME format)

Table 36. (Part 11 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
VICINITY	VHB	Bomber cannot go high within VHB miles before target
	VHA	Bomber cannot go high within VHA miles after target
	VLB	Bomber cannot go low within VLB miles before target
	VLA	Bomber cannot go low within VLA miles after target
	GOMIN	Bomber cannot fly low for less than GOMIN minutes
	DELDIS(6)	Decoy coverage distance
	LPRIORITY(20)	Possible decoy launch priority
	LMHT(90)	Possible Decoy Launch event number
	NDCYRQ(20)	Pointer to array DELDIS
	NPSLN	Number of possible decoy launches
	NUMDCOYS	Number of decoys available
	IWARFL	Logical unit number for war gaming print output
	COST(I,J)	Distance between tanker base I and refuel area J (where I=60, J=50)
	SOURCE(60)	Number of tankers at base I to be automatically assigned
	ISOL(110)	The Ith nonzero element in final VAM solution
/C9/	RBASLOC(110)	Tanker base corresponding to the Ith solution element
	CBASLOC(110)	Refuel area corresponding to Ith solution element
	NSOL	Number of nonzero elements in VAM solution
	RMAX	Number of rows (tanker bases) in VAM problem
	IRCHK(60)	Set to one if base I tankers are not to be automatically allocated
	IRCDIF	Number of bases for which IRCHK(I)=1
	DISTREF(50)	Distances from current tanker base to refuel area I

4.6.1 Overlay PLANO1

PURPOSE: This first overlay of program PLANOUT permits the user to make minor changes to the plans generated by QUICK without requiring a new allocation

ENTRY POINTS: PLANO1

FORMAL PARAMETERS: None

COMMON BLOCKS: BASEF, BEGIN, BLOCK, CARDS, CONTROL, CONTR1, CORRCHAR, CORRC1, DPENRE, FILABEL, FILES, ITP, MYIDENT, MYLABEL, NOPRINT, OUTSRA, OUTSRT, PAYDATA, PLNTHL, TARLIST, TWORD, WAROUT

SUBROUTINES CALLED: ABORT, CHGCORR, CHGSRT, DEACTIV, FLTSORT, ITCHG, LOCATE, MKCHG, RDARRAY, RDWORD, SETREAD, SETWRITE, SKIP, SLOG, TERMTAP, WRARRAY

CALLED BY: PLANOUT

Method:

The first sortie change card is read and if the operation code is an "E" there are no changes and the first overlay terminates. If not, the change cards are read, sorted by Sortie Sequence Number and saved on a temporary file. Then the first card is read from the file and the processing of all change cards begins.

The first phase is the initialization of input files and performance of the proper reads. TARFILE is initialized; the header is read into common /TARLIST/ and its file is terminated. Subroutine FINDTAR will reopen and close TARFILE as needed.

BASFILE parameters are now read into common /BASEF/, /CORRCHAR/ and /DPENRE/. /BASEF/ array contains information required if new missile sorties are added; other data are used in subroutine CHGSRT when bomber attrition and low-altitude distance are recalculated. If no STRKCHNG file exists, subroutine CHGCORR is called to fill required arrays for use in FLTSORT. Once defined, these arrays are placed on the STRKCHNG file and kept for future runs.

PLANO1 next determines the latest STRKCHNG file and defines it as the input file, SCHNG1. The various combinations are: no STRKCHNG exists, one or two STRKCHNG exist. Function LOCATE determines the existence of files and subroutine SETREAD stores the creation date and times into common /FILABEL/. Parameters INDSK and INDSKM define input file number and

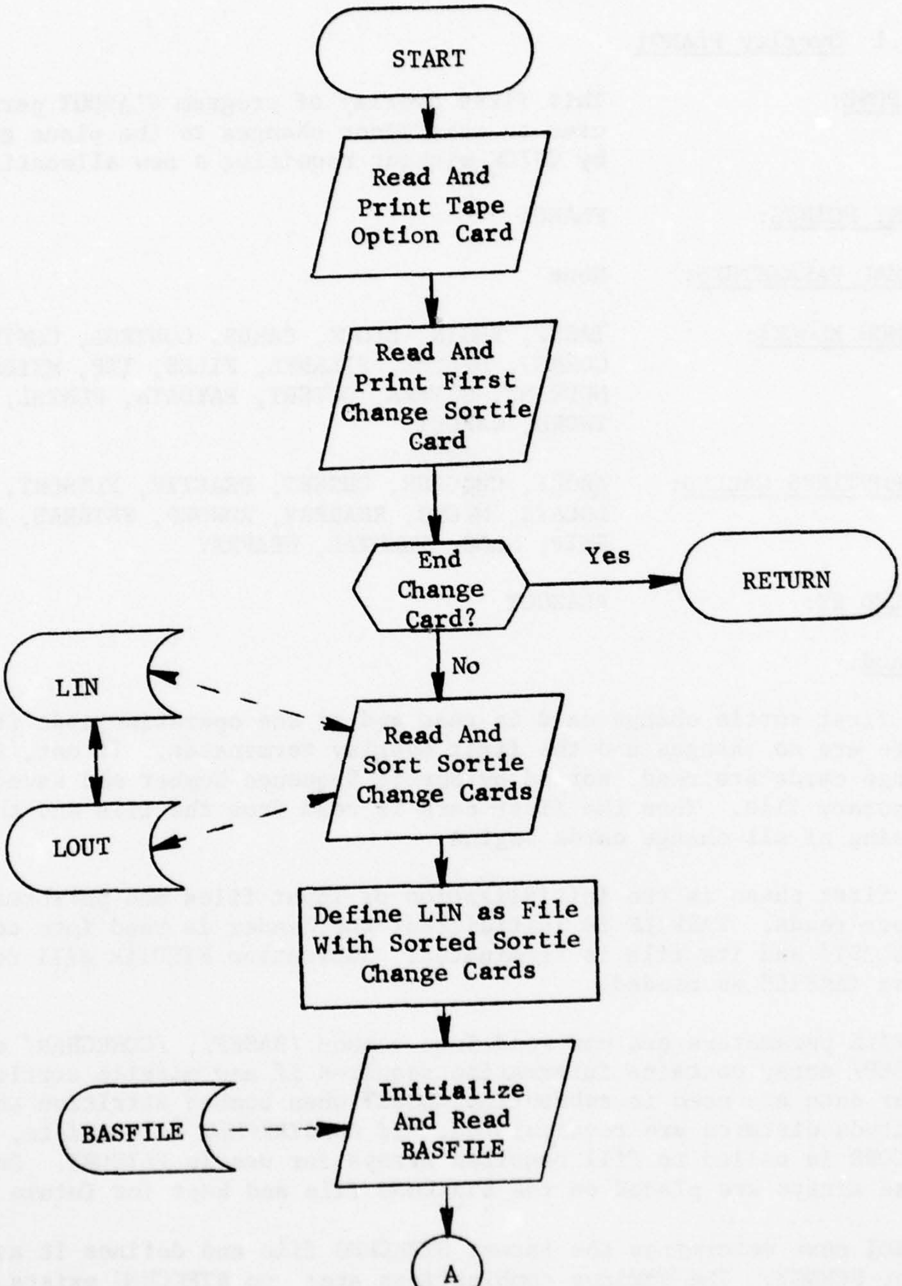


Figure 76. Overlay PLAN01
(Part 1 of 7)

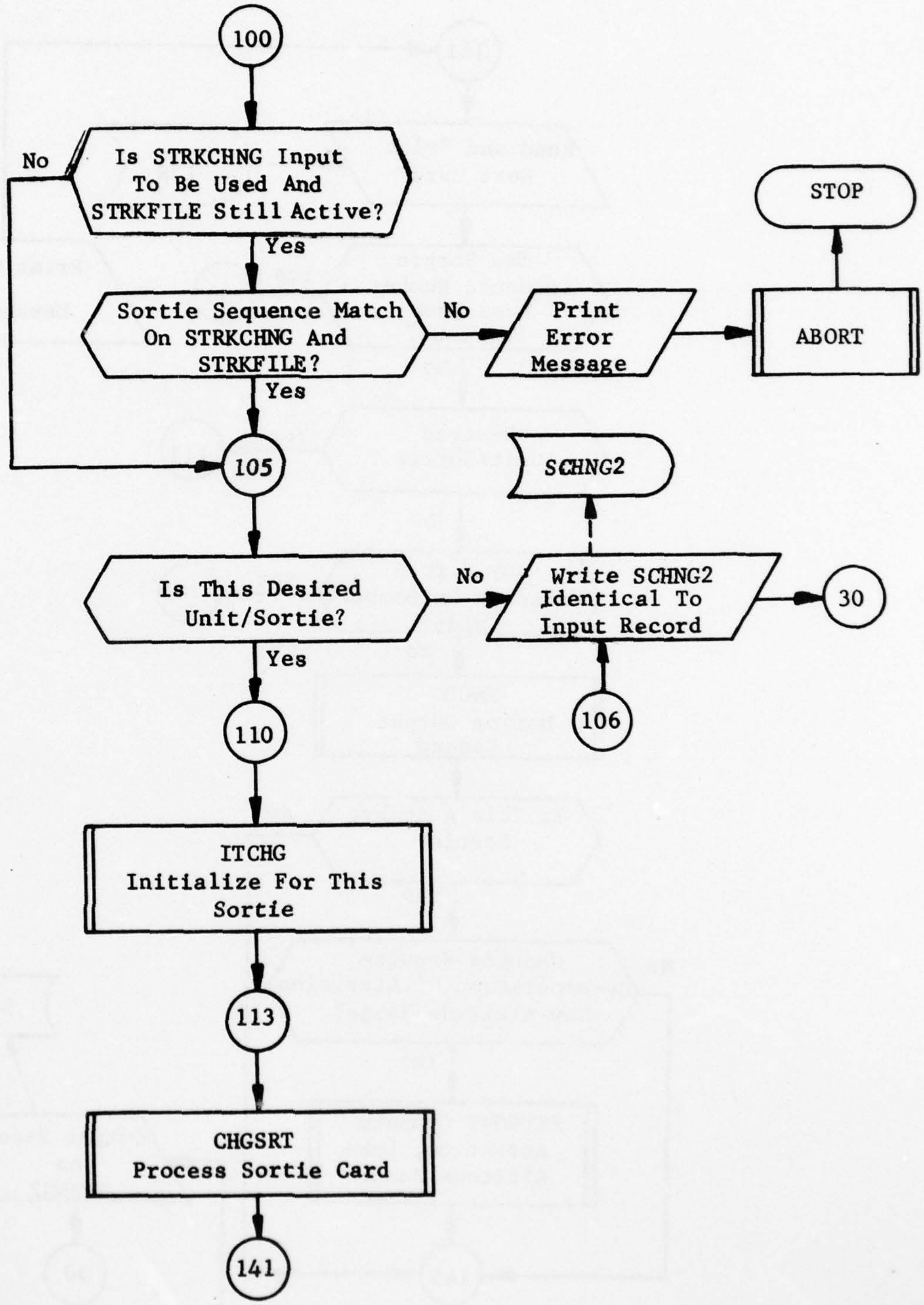


Figure 76. (Part 6 of 7)

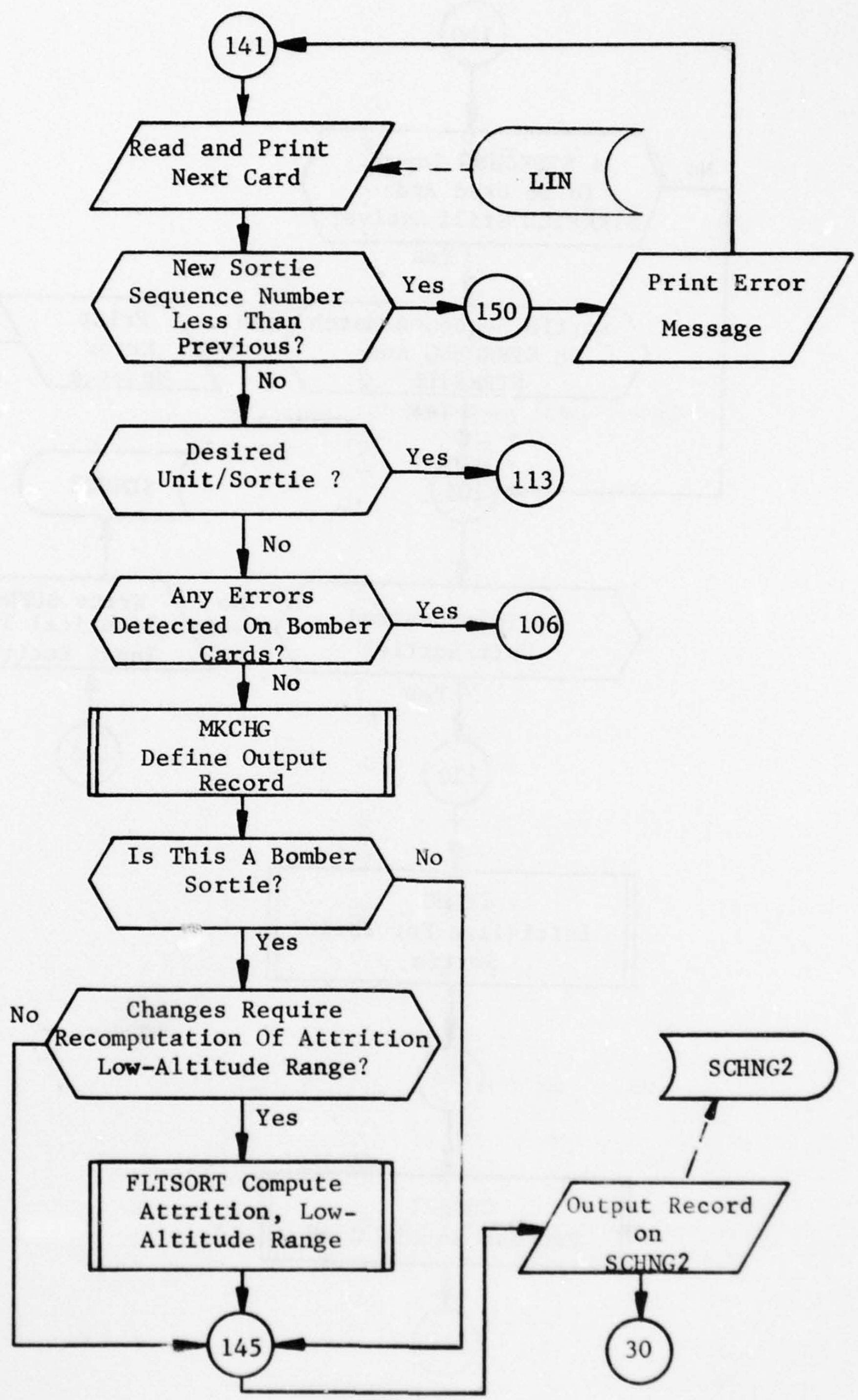


Figure 76. (Part 7 of 7)

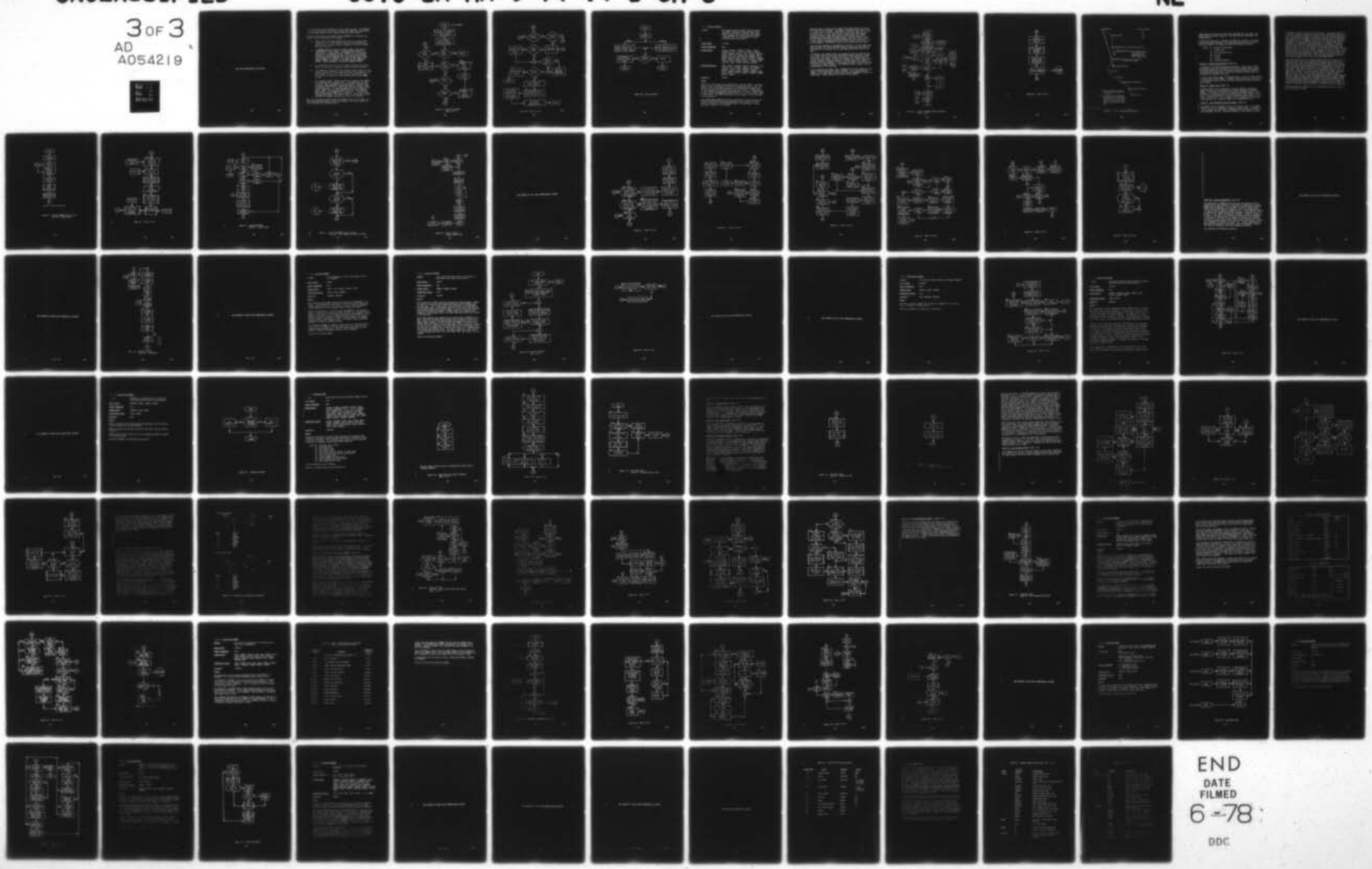
AD-A054 219 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 15/7

THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK). PROG--ETC(U)

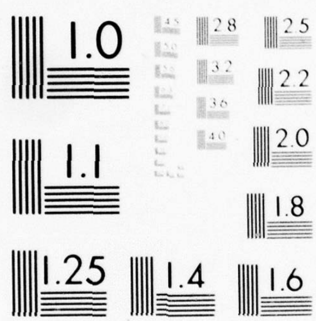
APR 78

UNCLASSIFIED CCTC-SM-MM-9-74-V4-1-CH-3 NL

3 OF 3
AD
A054219



END
DATE
FILMED
6-78
DDC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

THIS PAGE INTENTIONALLY LEFT BLANK

file name; IOUTDSK and IOUTDSKM the output number and name. The STRKCHNG header is read, if any file exists. Finally, the STRKFILE is opened and the first word TARFAC is read. SCHNG2 writes the output header.

Merging of input data with STRKFILE and/or STRKCHNG (if it exists) now begins. The cyclic process is as follows:

- Read a plan into common /OUTSRT/ and if it is a missile plan transfer the plan into /BLOCK/ array and finish reading the missile plan. If local parameter NOREAD1 is nonzero, STRKFILE has been terminated.

- A STRKCHNG record is read if it exists IST≠0. Each read is accomplished in three steps. A two-word header exists for each sortie. If the second word, LREAD(2), is zero the plan as given on the STRKFILE is used; if not this plan has been altered in previous PLANOUT runs. For this case the altered plan follows on the STRKCHNG. The plan is read, again, into common /OUTSRT/ or /BLOCK/, thus eliminating the STRKFILE record. Added information is read into common /OUTSRA/.

- It is imperative that the sortie sequence number on both files match for all plans. A mismatch causes the program to abort.

- The STRKCHNG (or STRKFILE) sortie sequence number, ISORTN, is now compared against the sortie sequence number, ISSN, defined on the last change card read. If ISSN is greater than ISORTN, the present plan is unaltered on this run and written onto SCHNG2 as read.

- For a change request, ITCHG is called to initialize; then CHGSRT to process the sortie change card. A new change card is read and sortie sequence numbers are again compared. Whenever ISSN is greater than ISORTN the present plan may be finalized; MKCHG accomplishes this. If any errors were detected on bomber sortie cards, ICHANGE <0, the output STRKCHNG will be the unchanged input record. Subroutine FLTSORT is used for bomber plan and when recalculation, ICHANGE >0, of attrition is desired. After meeting these checks, the altered SCHNG2 is written and the cycle continues by reading a new plan on STRKFILE and STRKCHNG.

This cyclic process continues until the STRKFILE end-of-file is read. At this stage non-MIRV missile sorties may be added. One sortie change completely defines a new missile sortie.

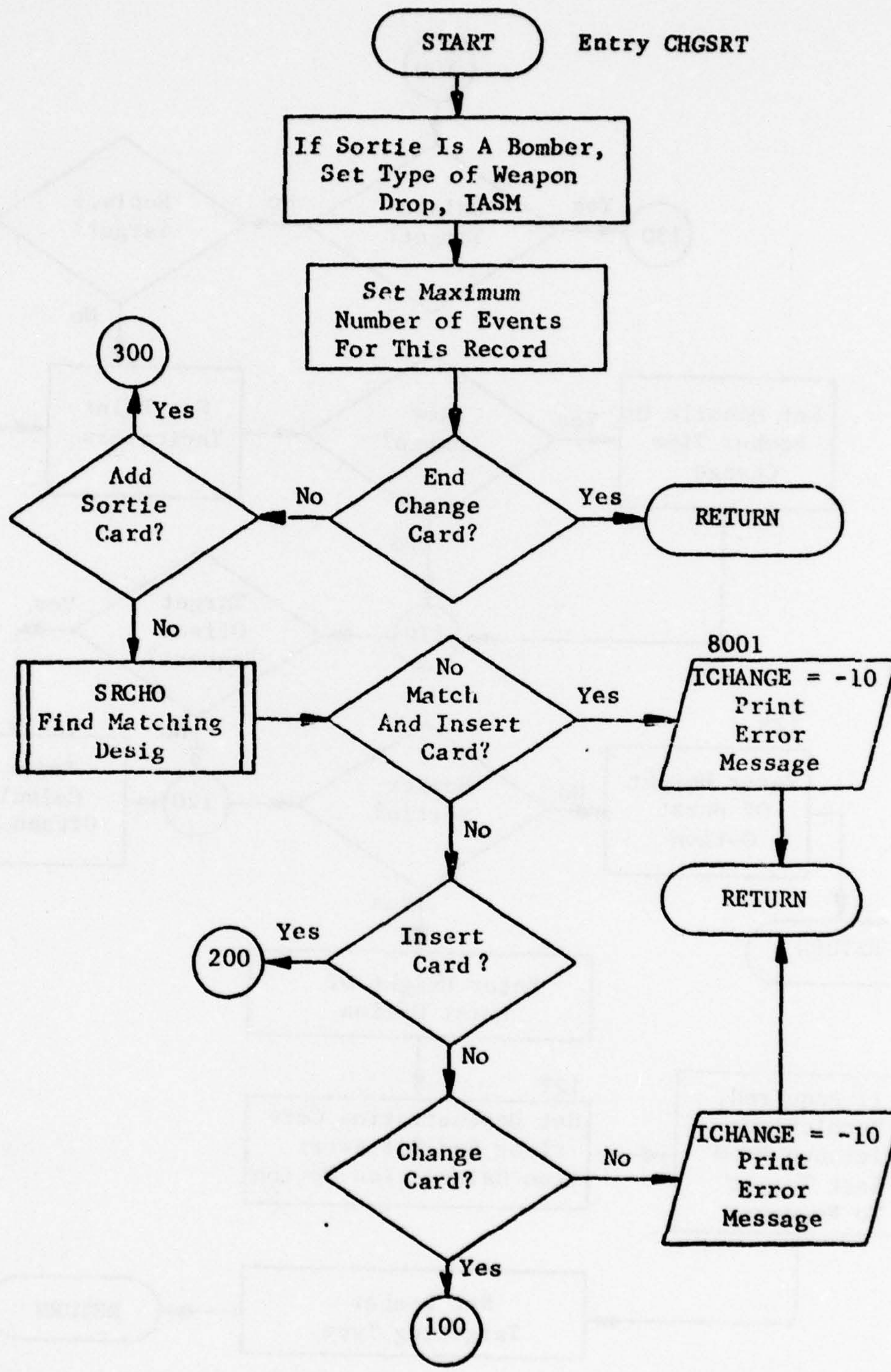


Figure 78. Subroutine CHGSRT
(Part 1 of 7)

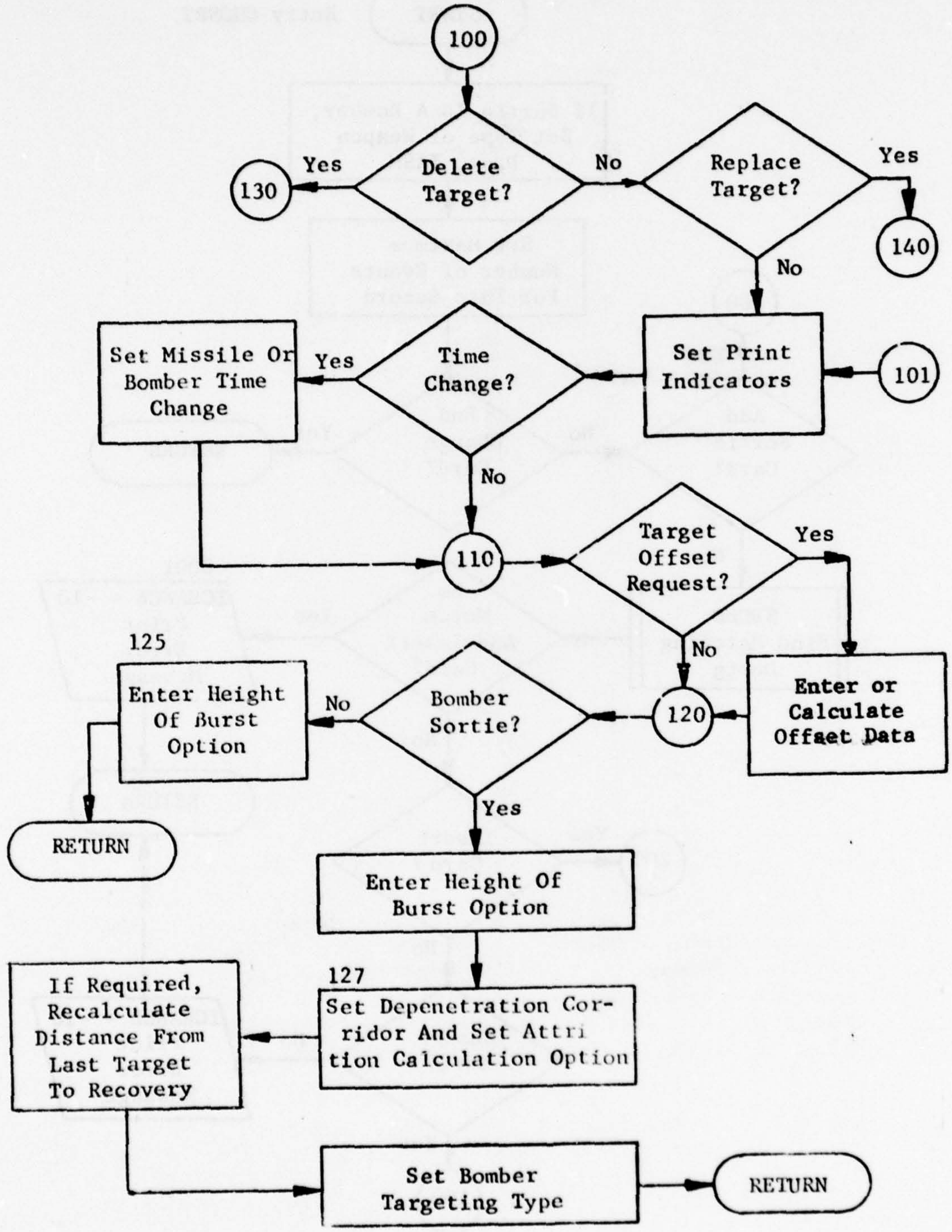


Figure 78. (Part 2 of 7)

CH-3

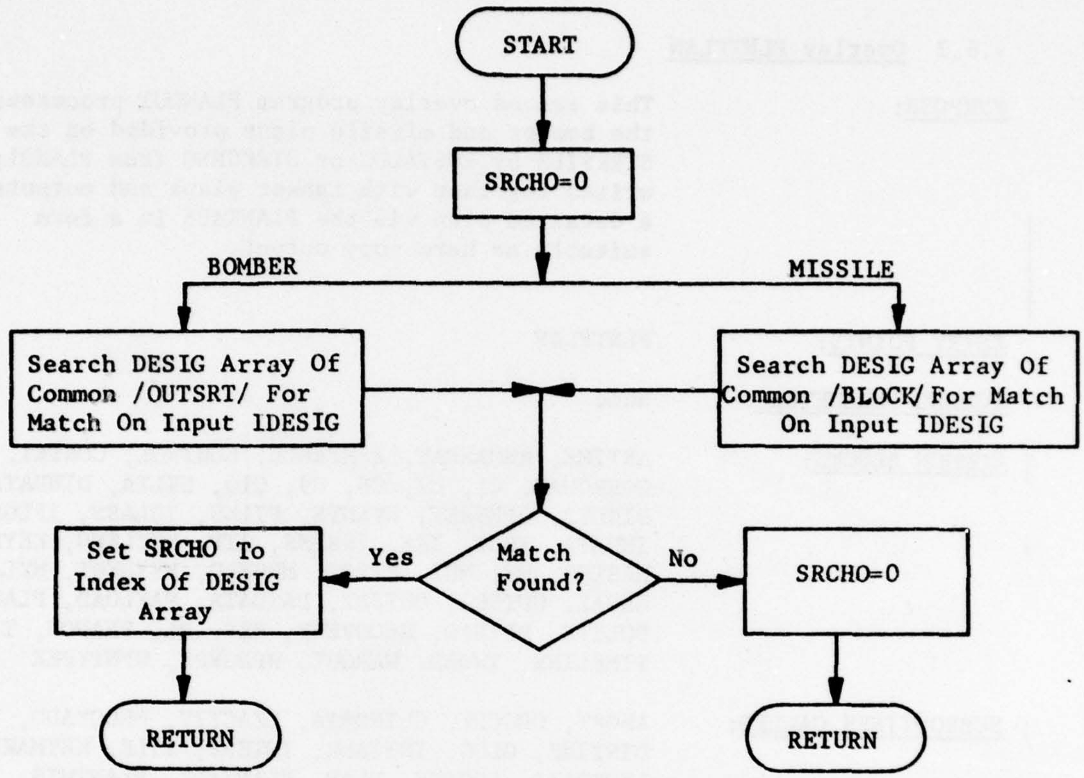


Figure 81. Function SRCHO

+.6.2 Overlay PLNTPLAN

PURPOSE: This second overlay program PLANOUT processes the bomber and missile plans provided on the STRKFILE by POSTALOC or STRKCHNG from PLAN01; writes together with tanker plans and outputs a detailed plan via the PLANTAPE in a form suitable as hard copy output.

ENTRY POINTS: PLNTPLAN

FORMAL PARAMETERS: None

COMMON BLOCKS: ARTIME, ASMARRAY, ASMTABLE, CONTROL, CONTR1, CORRCHAR, C1, C7, C8, C9, C10, DELTA, DINDATA, DINDT2, DPENREF, EVENTS, FILES, ICLASS, IFLGDPEN, INDATA, IOUT, IRF, ISRTNS, ITP, KEYLENG, KEYS, MASTER, MH, MH2, MISCT, MRVFLG, MYIDENT, MYLABEL, NAVAL, OUTSRA, OUTSRT, PAYDATA, PAYLOAD, PLANTYPE, POLITE, PPINFO, RECOVERY, REF, RL, SNAPON, TIME, TIMELINE, TWORD, WAROUT, WPNGRPX, WPNTYPEX

SUBROUTINES CALLED: ABORT, CHGCHK, CLINDATA, DEACTIV, DECOYADD, DISTF, DISTIME, GLOG, INITANK, INTERP, ITLE, KEYMAKE, LNCHDATA, LOCATE, PLAN, PLANTANK, PLANTMIS, PRNTAB, RDARRAY, RDWORD, SETREAD, SETWRITE, SKIP, SLOG, SNAPCON, SNAPIT, SWTCHALT, TERMTAP, TIMEME, WRARRAY, WRWORD

CALLED BY: PLANOUT

Method

Figure 82 is the overall macro flowchart for overlay PLNTPLAN. Initialization includes reading the BASFILE and the user data cards. Also, the STRKFILE first word, TARFAC, is read, and STRKCHNG header skipped over. The first record from the STRKFILE is read in, subroutine CHGCHK called to read STRKCHNG and the main processing begins. Whenever a missile plan is read, subroutine PLANTMIS is given control until another bomber record is encountered. (A bomber record with a group number of 999 is the end-of-file indicator.)

Since subroutine PLANTMIS controls all processing of missile plans, and subroutine PLANTANK controls the generation of all tanker plans, the relevant concepts are discussed under those two subroutines. Bomber processing, however, is as follows.

Figure 83 shows a typical path a bomber would take between the time of its launch and its recovery. The bomber is launched from a base, flies to a refuel point or area if refueling is called for, then to a corridor entry point. It may then fly one or more prespecified doglegs (called precorridor legs) which define a penetration route before reaching the point labeled Corridor Origin. From the origin it flies over the target area and its assigned targets in their proper order. It then enters the depenetration corridor which may also consist of one or more doglegs. From there it flies to the recovery point or base.

This path may logically be divided into four parts: (1) the launch and refuel portion, (2) the precorridor legs, (3) the target area which is the main part of the plan, and (4) the depenetration and recovery portion.

In PLNTPLAN, each bomber sortie is processed in much the same order as it is flown; that is, first the precorridor section events are posted, then those of the target section, and finally, the depenetration and recovery section events. Besides the posting of the target events themselves, the main processing consists of posting events for changes of altitude and decoy launches. All postings for bomber events are made in the arrays of common /DINDATA/. The completed plan is output, and processing begins on the new plan.

After processing all bomber plans, STRKCHNG must be investigated for any missile sorties added by the user in PLAN01. If the sortie sequence number is less than 99999, sorties were added and PLANTMIS processes all added plans. These sorties do not exist on the STRKFILE.

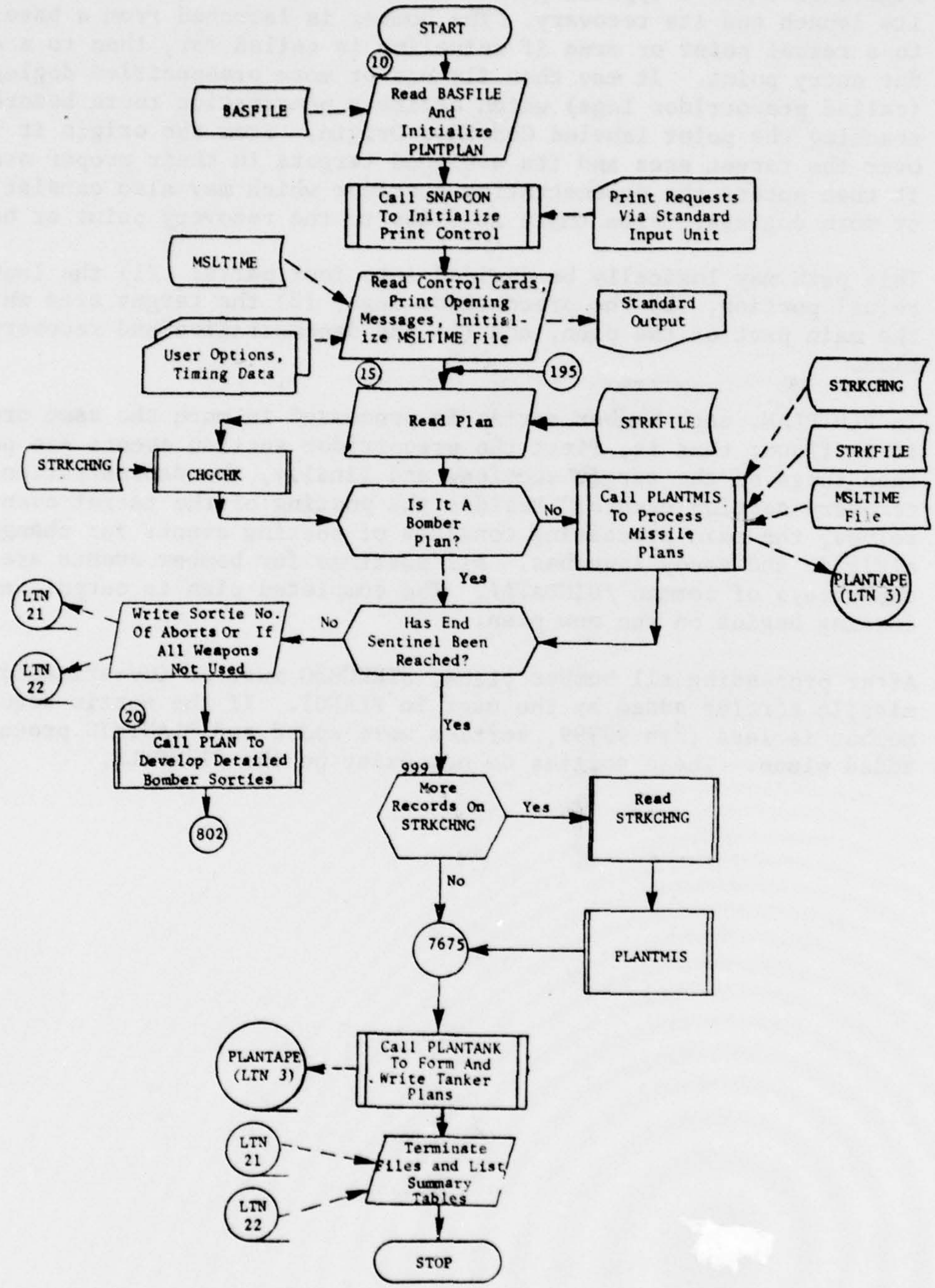


Figure 82. Overlay PLNTPLAN (Macro Flowchart) (Part 1 of 2)

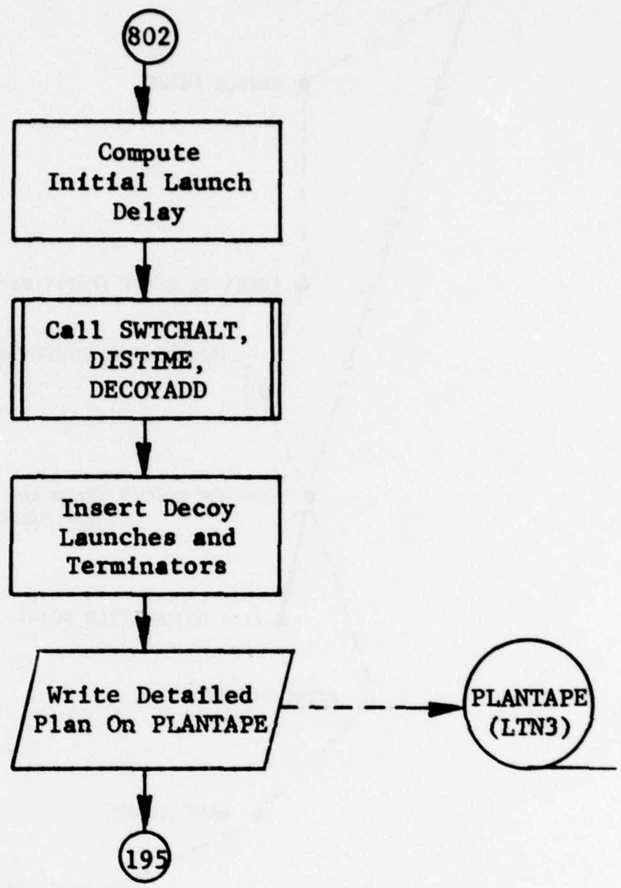


Figure 82. (Part 2 of 2)

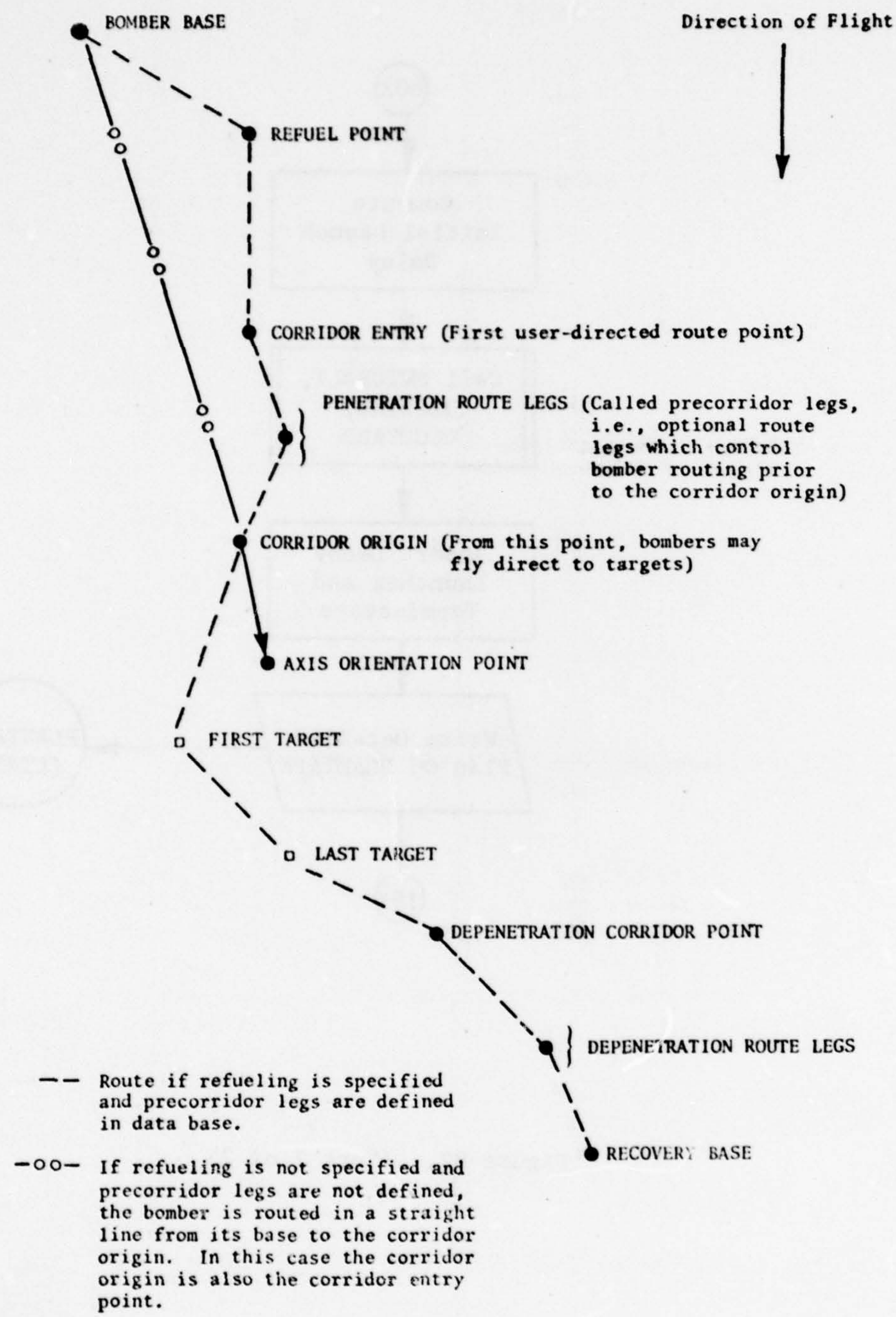


Figure 8[illegible]. Path of Typical Bomber Sortie

Tanker plans are generated by subroutine PLANTANK after all bomber and missile plans have been completed. All files are then terminated, and PLNTPLAN exits.

To facilitate discussion, PLNTPLAN is divided into "blocks" of coding as noted in the macro flowchart. The remaining description as well as the detailed flowcharts are organized around these blocks, which are:

BLOCK	10	- Program initialization
	15	- Control loop
	20	- Call subroutine PLAN
	80	- Deleted
	90	- Deleted
	100	- Program termination

Block 10: Program Initialization (figure 84)

In addition to initialization program variables, coding block 10 reads all required data from the BASFILE, and all user control cards, calling subroutine SNAPCON for the print request cards and subroutine LNCHDATA for the missile timing cards. Files are initialized, headers read, and preliminary information is printed.

A list of each recovery base (determined by name as stored in array INDBAS) is stored within array NAMCAP. A single recovery base may be linked to up to four depenetration corridors and, hence, appear more than once within array INDBAS.

Block 15: Control Loop (figure 85)

The first bomber plan is then read in from the STRKFILE, subroutine CHGCHK executed to read STRKCHNG and the main processing of PLNTPLAN begins. If a missile plan is read, subroutine PLANTMIS receives control. Otherwise, the number of events is checked and the sortie INDEX is computed. Subroutine SNAPCON is called to determine which prints are to be active for this plan. If the program is to be terminated at this sortie (print request 14), a branch is taken to the termination block.

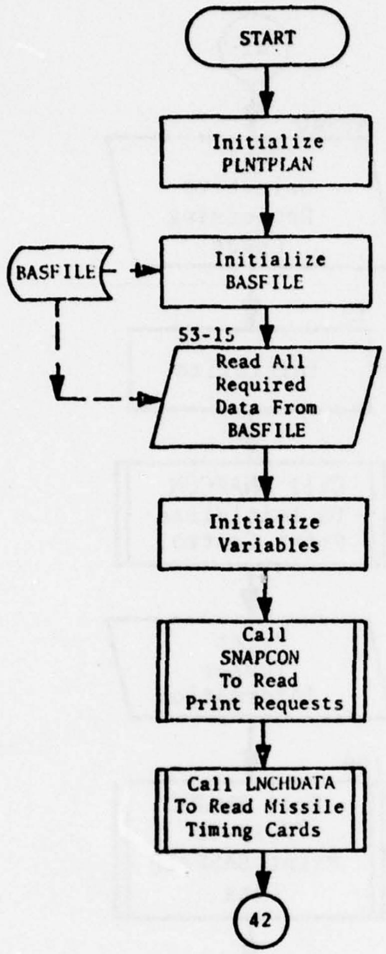
Block 20: Call Subroutine PLAN and Convert (figure 87)

The bomber sortie is checked to see if it is able to fly to a recovery base or if its full compliment of weapons were allocated. If processing is to continue with this record, subroutine PLAN is called to develop detailed bomber sorties. Subroutine SWTCHALT is then called to convert

the CHANGALT events to GO LOW or GO HIGH events. Subroutine DISTIME then is called to compute distances between events and associated time increments, and subroutine DECOYADD is called to allocate the available decoys. Decoy Launches are now added to the detailed History table by examining each event to see if a launch is to be inserted (indicated if the corresponding word in array ILAUNDEC is nonzero). For low-altitude launches (ILAUNDEC = 0), the actual launch point must be computed. The Decoy Launches are inserted by copying each event into a temporary detailed History table. If a GO HIGH event has a decoy launch indicated, the launch is inserted after the GO HIGH. For all other events with indicated decoy launches, the launch is inserted before the event is copies. Decoy launches are posted by adding to event LAUNDCOY to the event array (JTP) and storing the number of decoys launched (>0) in the array usually reserved for the place index (KPL). The remaining information required in the detailed History table is stored in the normal manner.

Decoys are terminated as the detailed History table is recopied into its original arrays. Each time a high-altitude Decoy Launch event is encountered, the total decoy flight time is computed from the distance in array DISTORE (filled by subroutine DECOYADD) and added to the next odd word in an array (TSTORE) which holds the remaining flight time of all decoys which have been launched but not yet terminated at the time of this event. The number of decoys to be terminated is added to the next even word of TSTORE. As each subsequent event is processed, the time since the last event (HDT) is subtracted from the times in TSTORE. Whenever a decoy has no flight time remaining, a LAUNDCOY event, together with the number of decoys being terminated (stored as a negative number) and other relevant information, is added to the detailed History table. If the bomber depenetrates or aborts while decoys are still flying, the remaining decoys are terminated immediately before the final event. It should be noted that decoys launched at low altitude are not terminated.

PLNTPLAN outputs the final plan to the PLANTAPE and returns to block 15 to obtain the next record.



Block 10: Program Initialization

Figure 84. Overlay PLNTPLAN (Part 1 of 2)
Block 10: Initialization

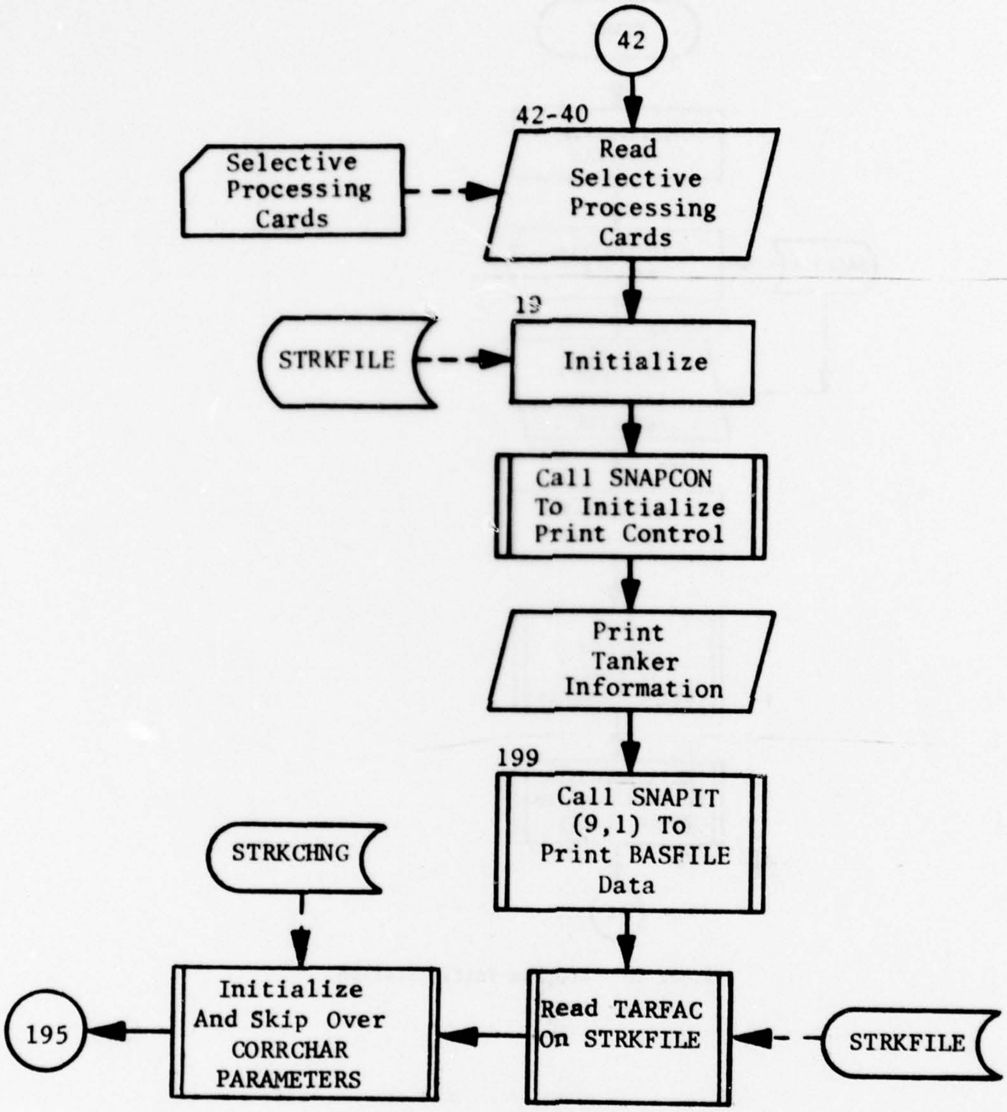


Figure 84. (Part 2 of 2)

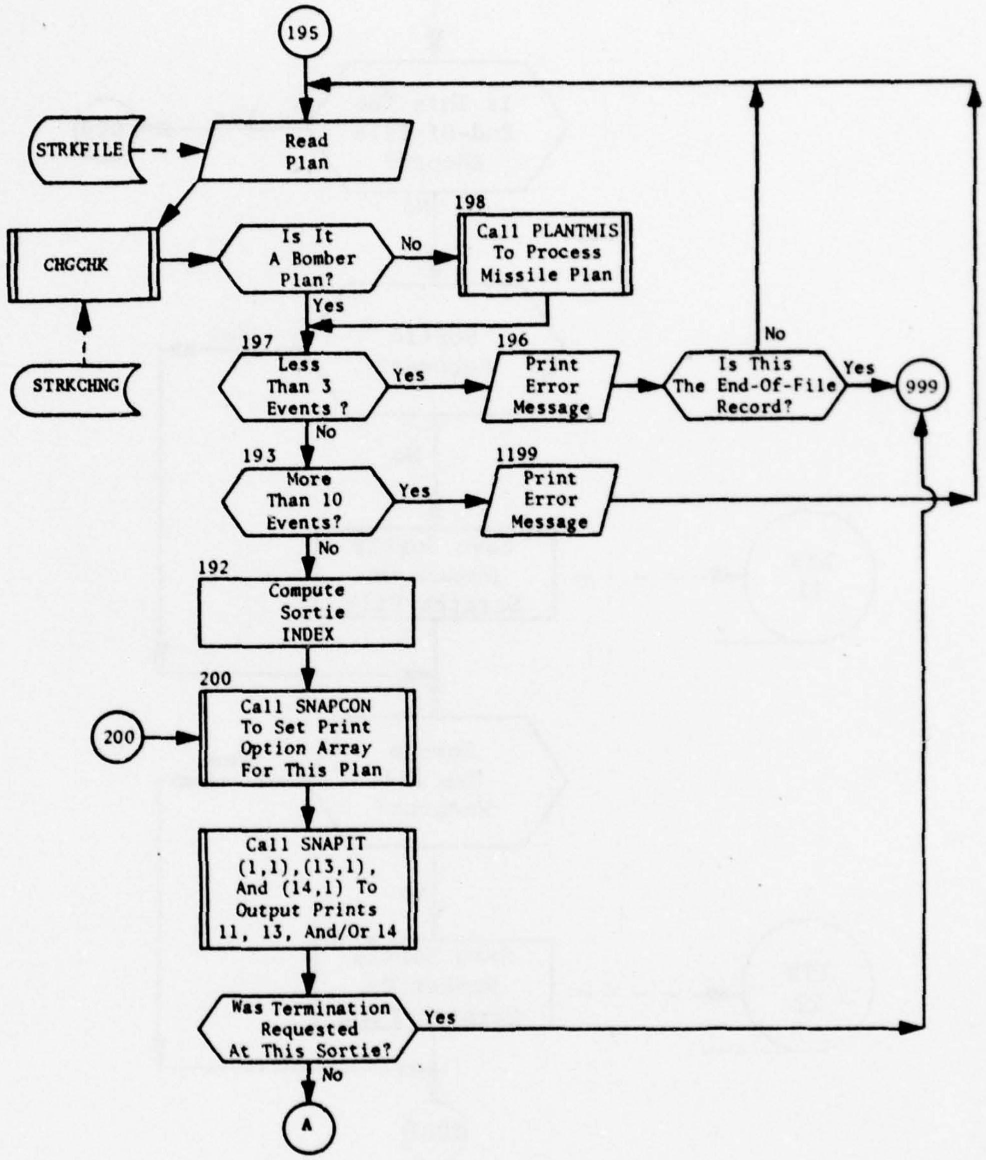


Figure 85. Overlay PLNTPLAN
Block 15: Control Loop

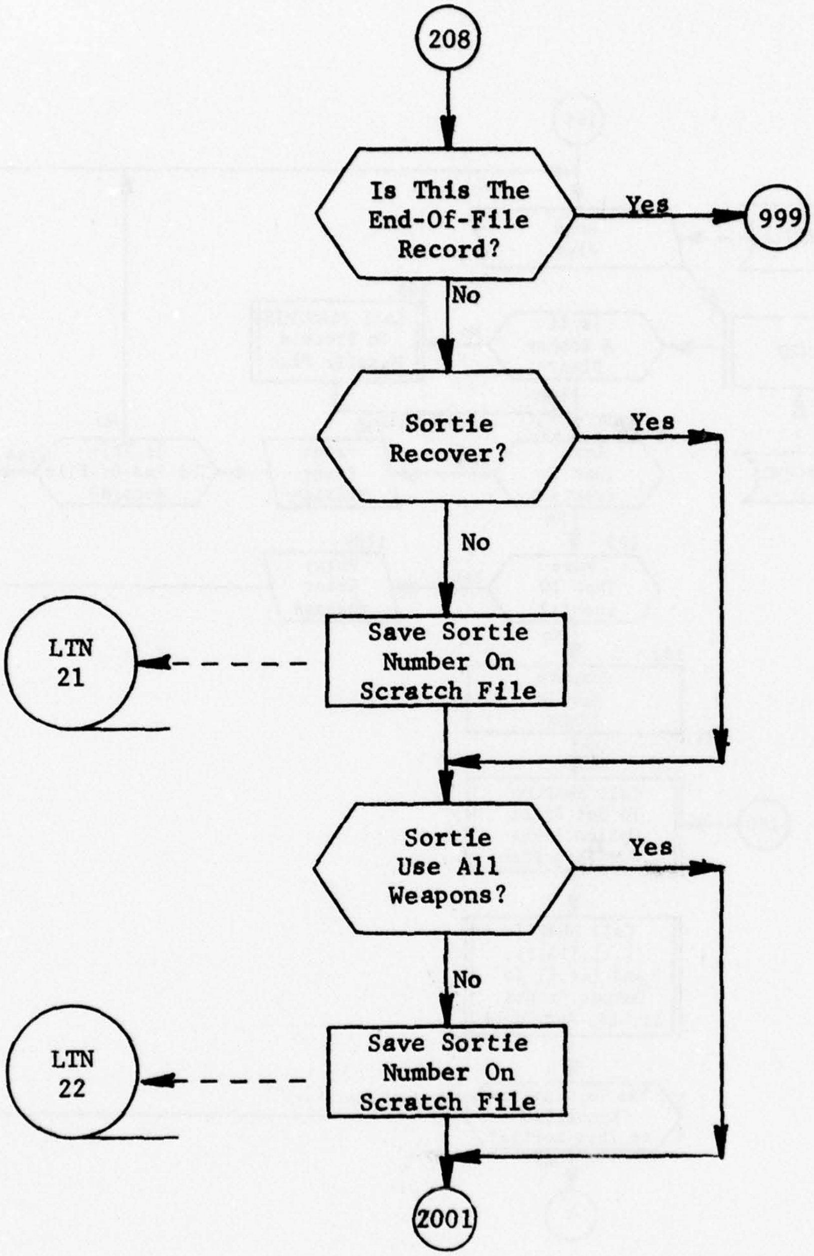


Figure 86. Overlay PLNTPLAN (Part 1 of 10)
Block 20: Call Subroutine PLAN and Convert

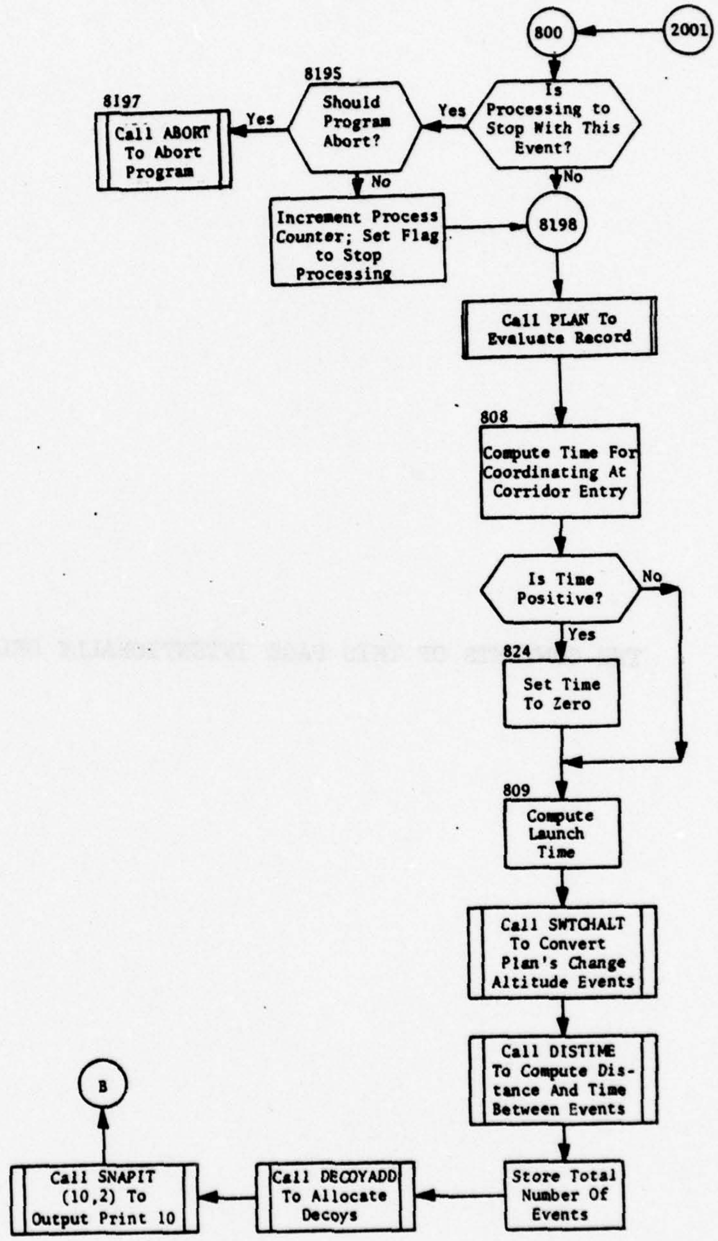


Figure 87. (Part 2 of 10)
Convert Bomber Plan

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

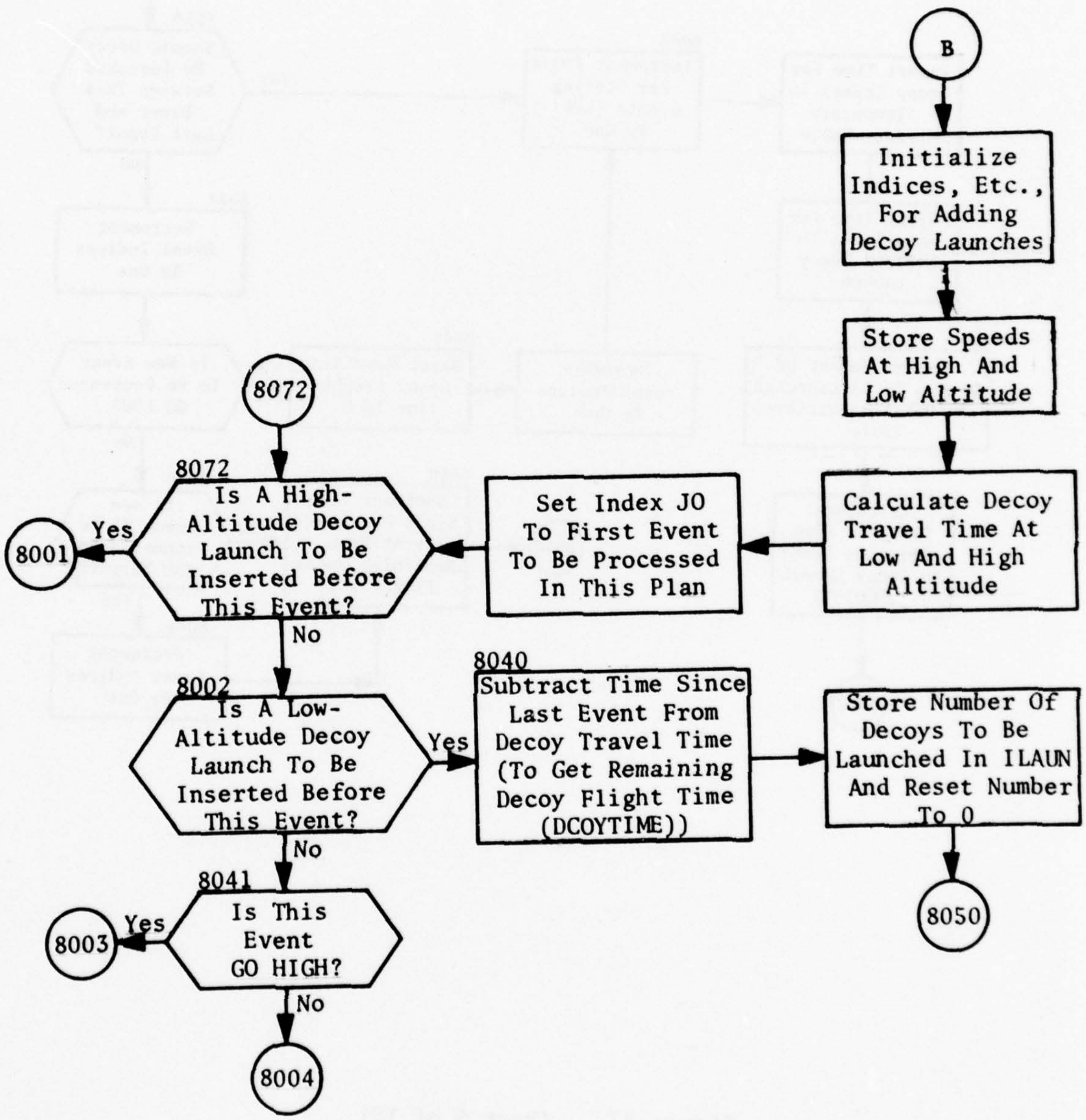


Figure 87. (Part 3 of 10)

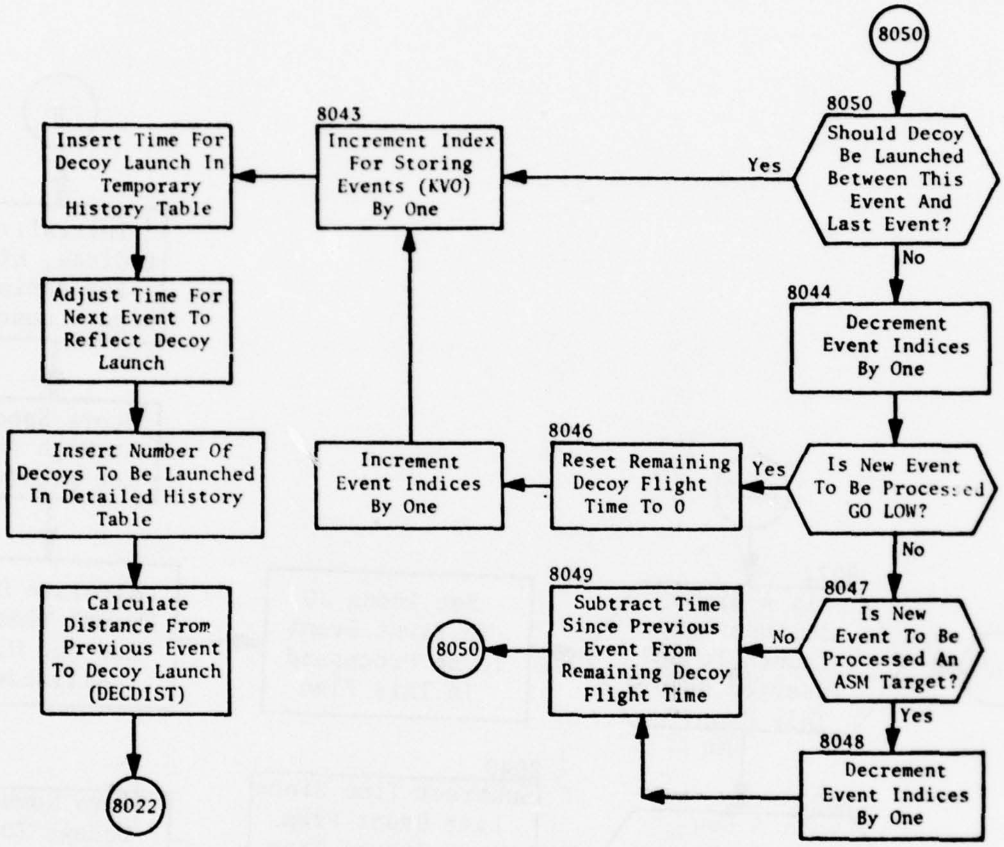


Figure 87. (Part 4 of 10)

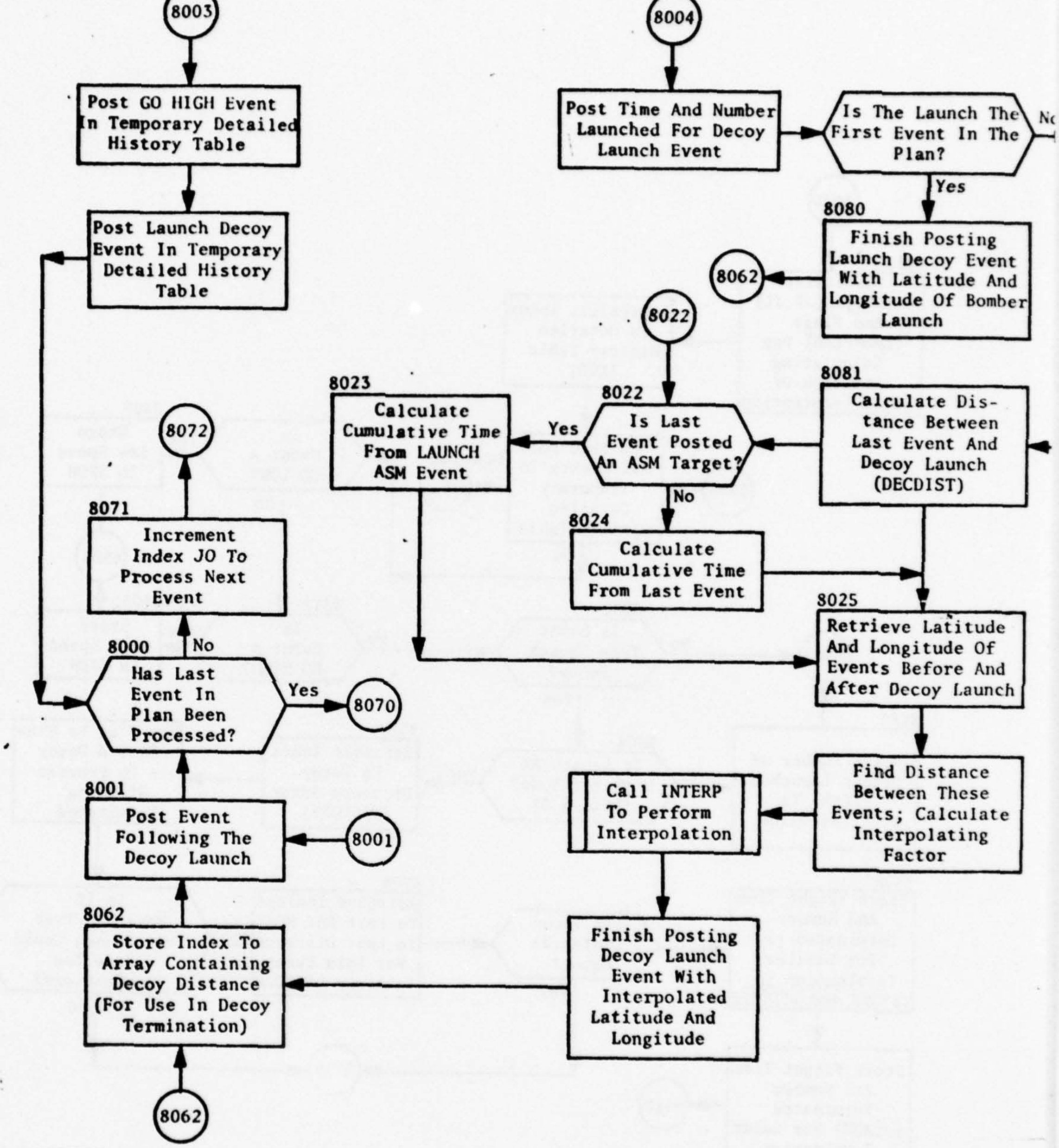


Figure 87. (Part 5 of 10)

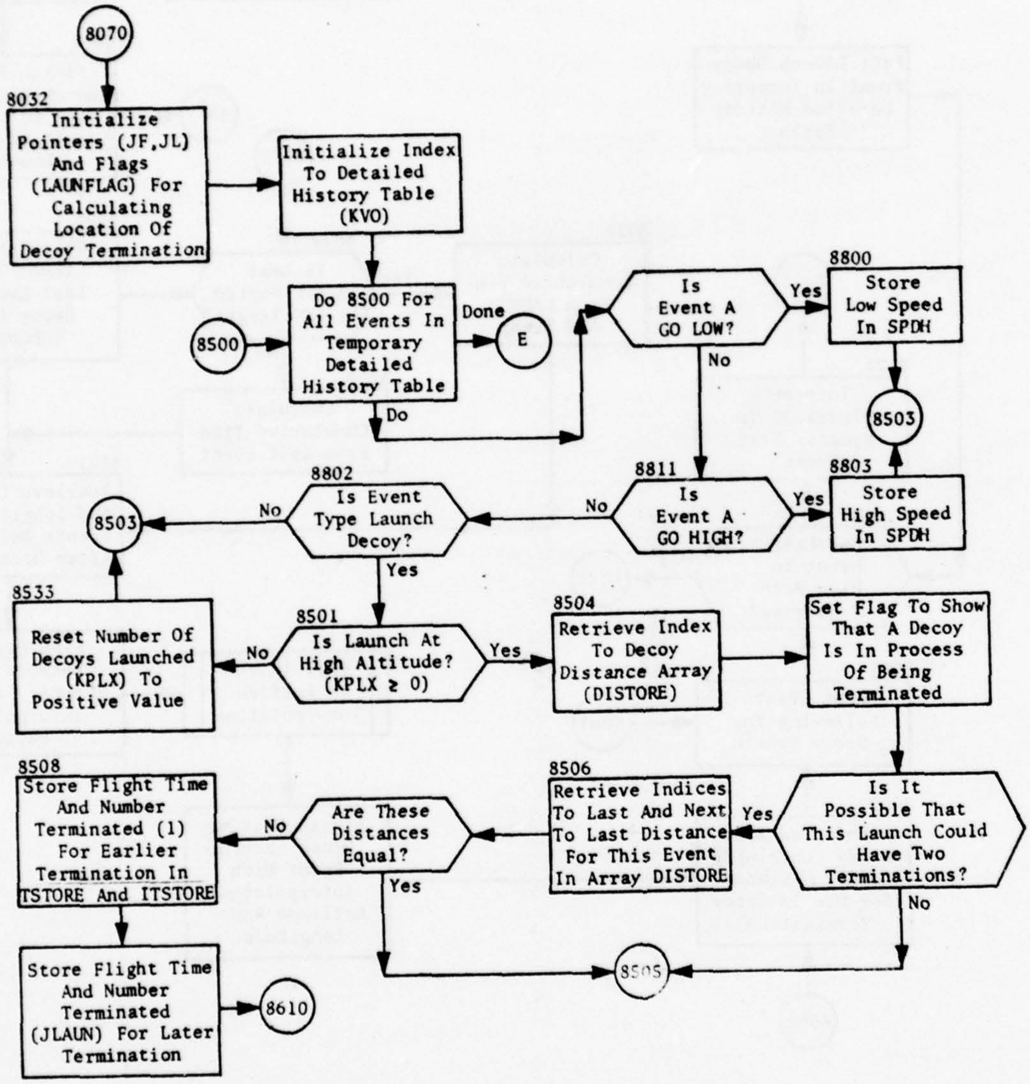


Figure 87. (Part 6 of 10)

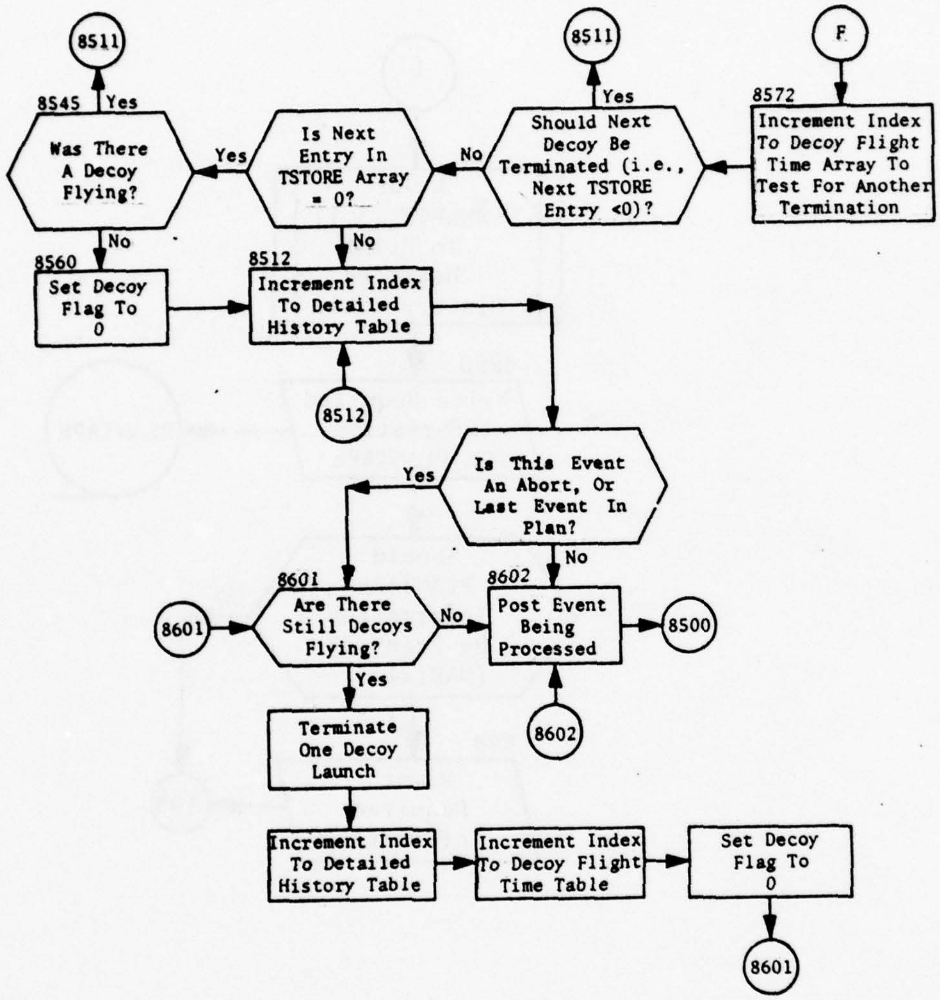


Figure 87. (Part 9 of 10)

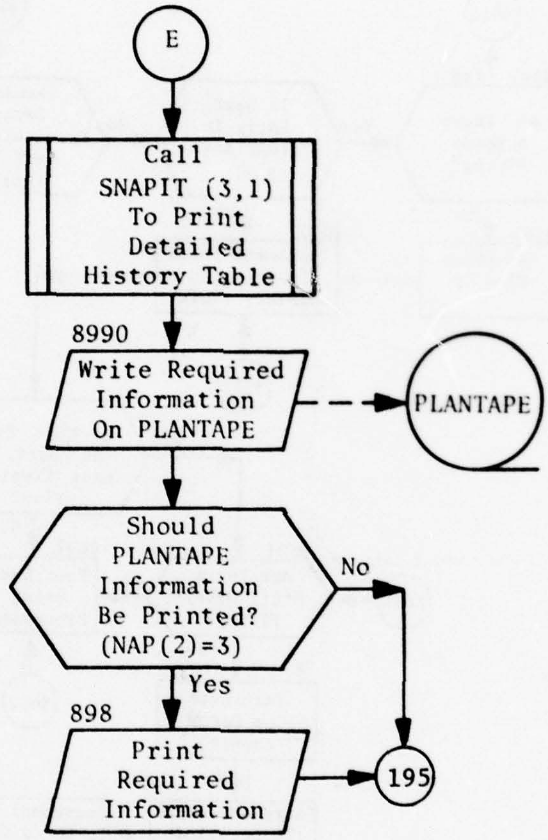


Figure 87. (Part 10 of 10)

Block 100: Overlay Termination (figure 89)

The termination of PLNTPLAN occurs when the end sentinel record is reached on the input STRKFILE. This record is identified by a group number greater than 250. At this stage all missile sorties added in PLAN01 onto the STRKCHNG are processed. A sortie sequence number greater than 99999, implies no added sorties. After PLANTMIS evaluates all added plans, subroutine PLANTANK is called to generate and write sorties for all tankers listed on the BASFILE. A sentinel record is written at the end of the PLANTAPE; the bomber recovery information is added to the eventape and the Refuel Area table to the PLANTAPE. At this point, the list of Sortie Numbers of the bombers that could not reach a recovery base and the list of those that did not use their full compliment of weapons, is printed out. A print summarizing how many aircraft arrives at each recovery base is generated. Finally, all files are terminated, and final messages are printed.

This completes the PLNTPLAN processing.

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

THE CONTENTS OF THESE PAGES INTENTIONALLY DELETED

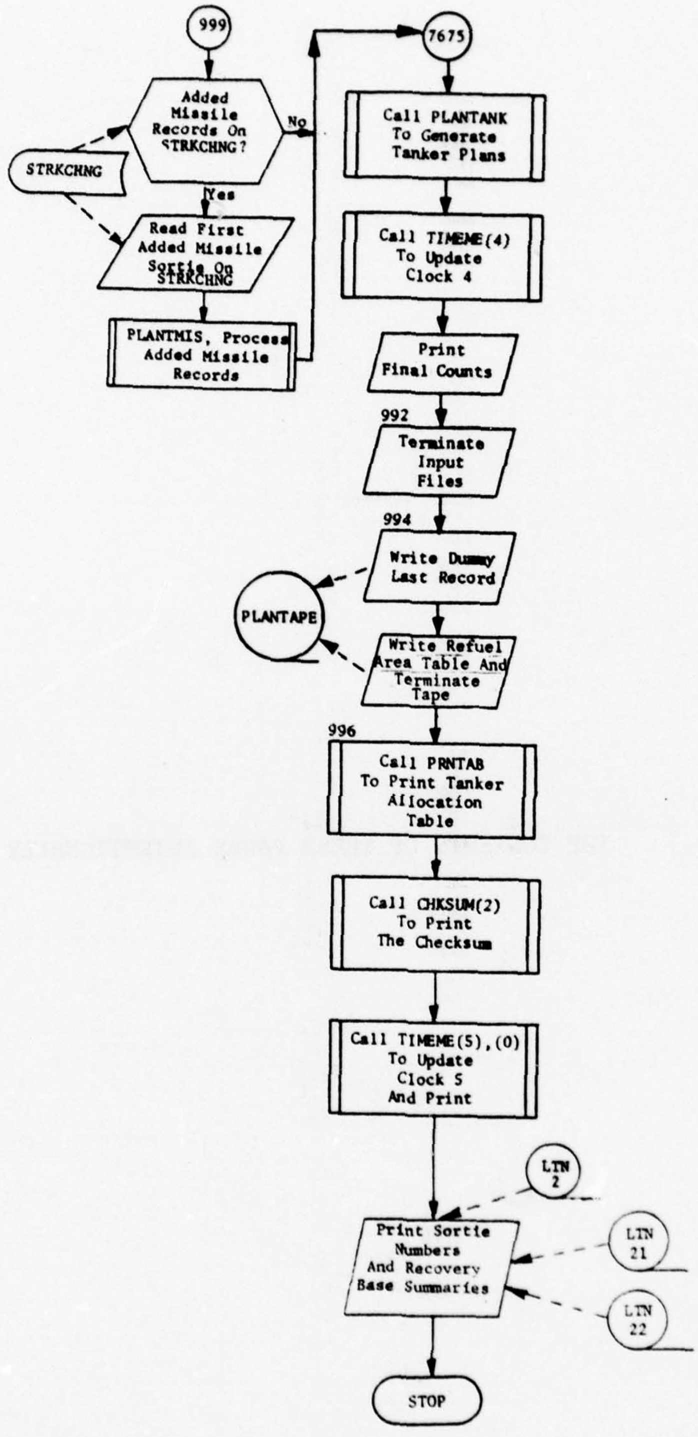


Figure 89. Overlay PLNTPLAN
Block 100: Termination

THE CONTENTS OF THESE PAGES INTENTIONALLY DELETED

4.6.2.3 Subroutine CHGCHK

PURPOSE: To read STRKCHNG as output from PLAN01, overlay 1 of PLANOUT.

ENTRY POINTS: CHGCHK

FORMAL PARAMETERS: None

COMMON BLOCKS: CONTR1, ITP, MYIDENT, OUTSRA, OUTSRT

SUBROUTINES CALLED: ABORT, LOCATE, RDARRAY

CALLED BY: PLANTMIS, PLNTPLAN

Method:

This subroutine reads bomber and missile plans from the STRKCHNG, if it exists. A two-word header consisting of the sortie sequence number and change indicator exists for each record. The last record on STRKCHNG is identified with a sortie sequence number greater than 99999.

CHGCHK is called after a record was read from STRKFILE. The two-word header is read first and checks are made to ensure sortie sequence number matches on both files. If the change indicator on the STRKCHNG is zero a RETURN is executed; otherwise a change record follows for this sortie sequence number. The record is read into common /OUTSRT/ and /OUTSRA/ whereby destroying the STRKFILE record. For missile plans, subroutine PLANTMIS will read the remaining record.

When parameter ICHANGE is negative, records are to be processed that do not exist on the STRKFILE. These records are missile plans that were added via sortie change cards. The first read of add sorties is performed in PLNTPLAN; subsequent reading is done in CHGCHK.

Figure 98 illustrates CHGCHK.

4.6.2.4 Subroutine CHGTIM

PURPOSE: Alter finalized history table by increasing or decreasing times among desired events.

ENTRY POINTS: CHGTIM

FORMAL PARAMETERS: None

| COMMON BLOCKS: CONTR1, DINDATA, OUTSRA

SUBROUTINES CALLED: None

CALLED BY: DISTIME

Method:

After subroutine DISTIME converts distances into time increments, CHGTIM is called to process all time changes specified on sortie change cards for this plan. Sortie cards permit time alterations to any DROPBOMB or AIM ASM weapon event. Interceding events will have these times adjusted by an amount proportional to their weight in the sum of the time between events as calculated by DISTIME. In other words, the requested time change for a given weapon event is prorated starting with the last DROPBOMB or AIM ASM event.

Array CHTIM contains time changes for all weapon events (DROPBOMB or AIM ASM). Array HDT contains time increments for all events in the sortie. These events consist of the weapon events plus additional events such as LAUNCH, REFUEL, DOGLEG, GO LOW, GO HIGH, and RECOVER. Also there is an ASM TGT event that defines an ASM flight time. The two arrays are similar in that the first weapon event in array CHTIM and the first weapon event in array HDT are identical as are the second, third, etc. CHGTIM prorates time changes among weapon events. The time increment for event ASM TGT is never changed since this is the ASM flight time; only launch times are changed.

Figure 99 illustrates CHGTIM.

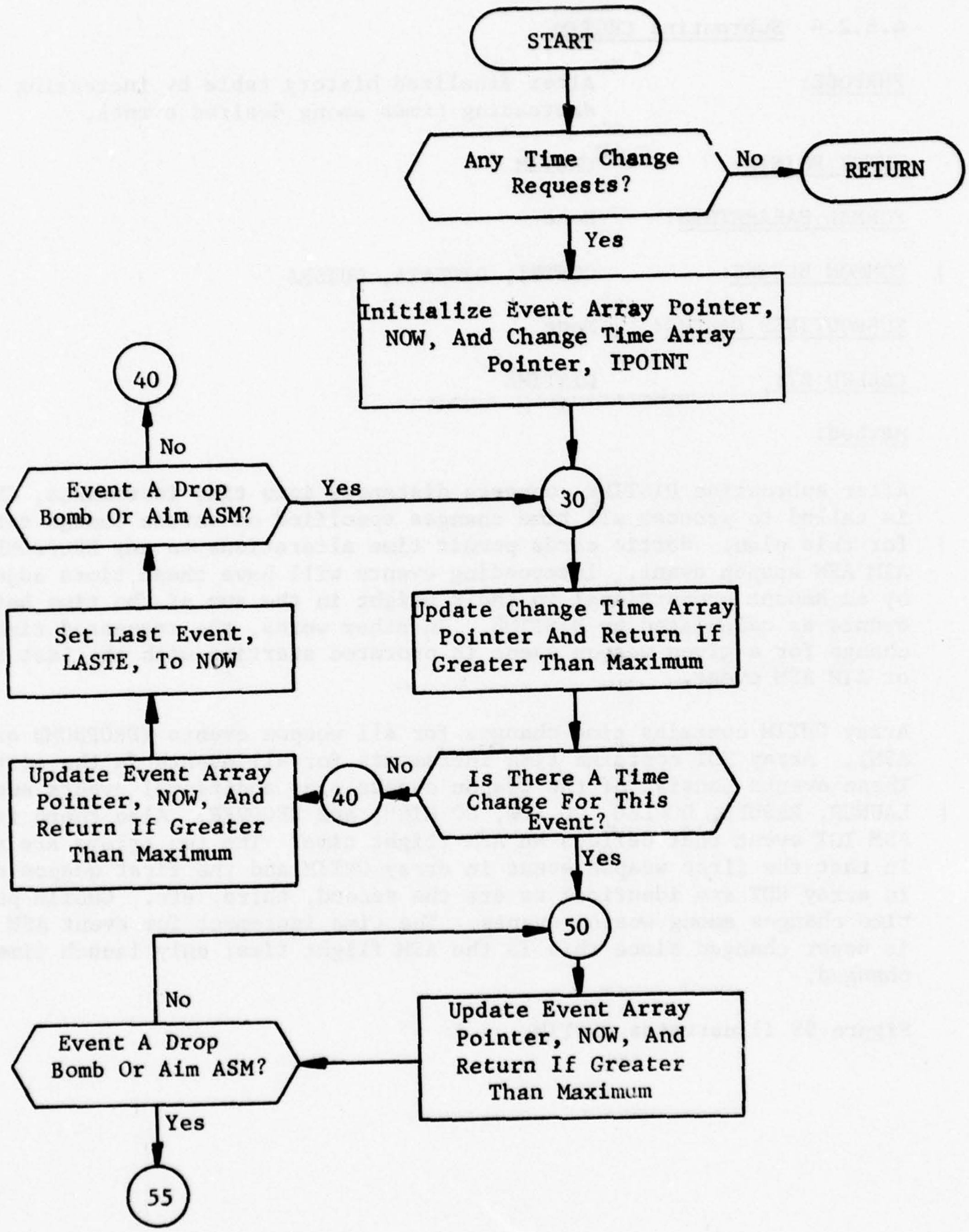


Figure 99. Subroutine CHGTIM
(Part 1 of 2)

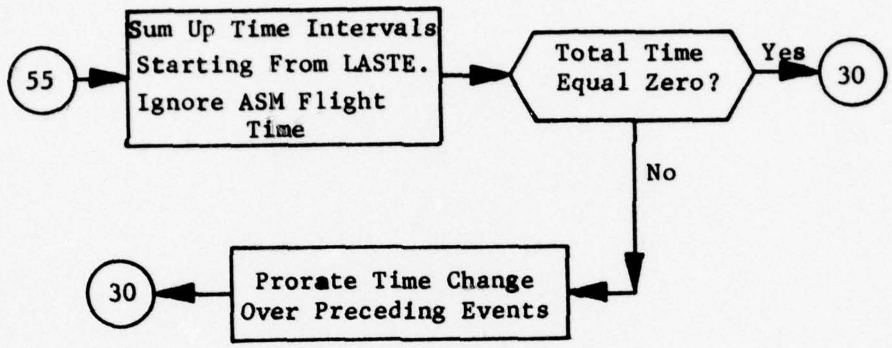


Figure 99. (Part 2 of 2)

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

4.6.2.6 Subroutine CLINDATA

PURPOSE: To initialize common /INDATA/ and common /DINDATA/

ENTRY POINTS: CLINDATA

FORMAL PARAMETERS: None

COMMON BLOCKS: DINDATA, INDATA, KEYLENG

SUBROUTINES CALLED: None

CALLED BY: PLAN, PLANTANK, PLNTPLAN

Method:

Each word in common /INDATA/ and each word in /DINDATA/ is set to zero. IALT in /INDATA/ is initialized to 1.

Subroutine CLINDATA is illustrated in figure 101.

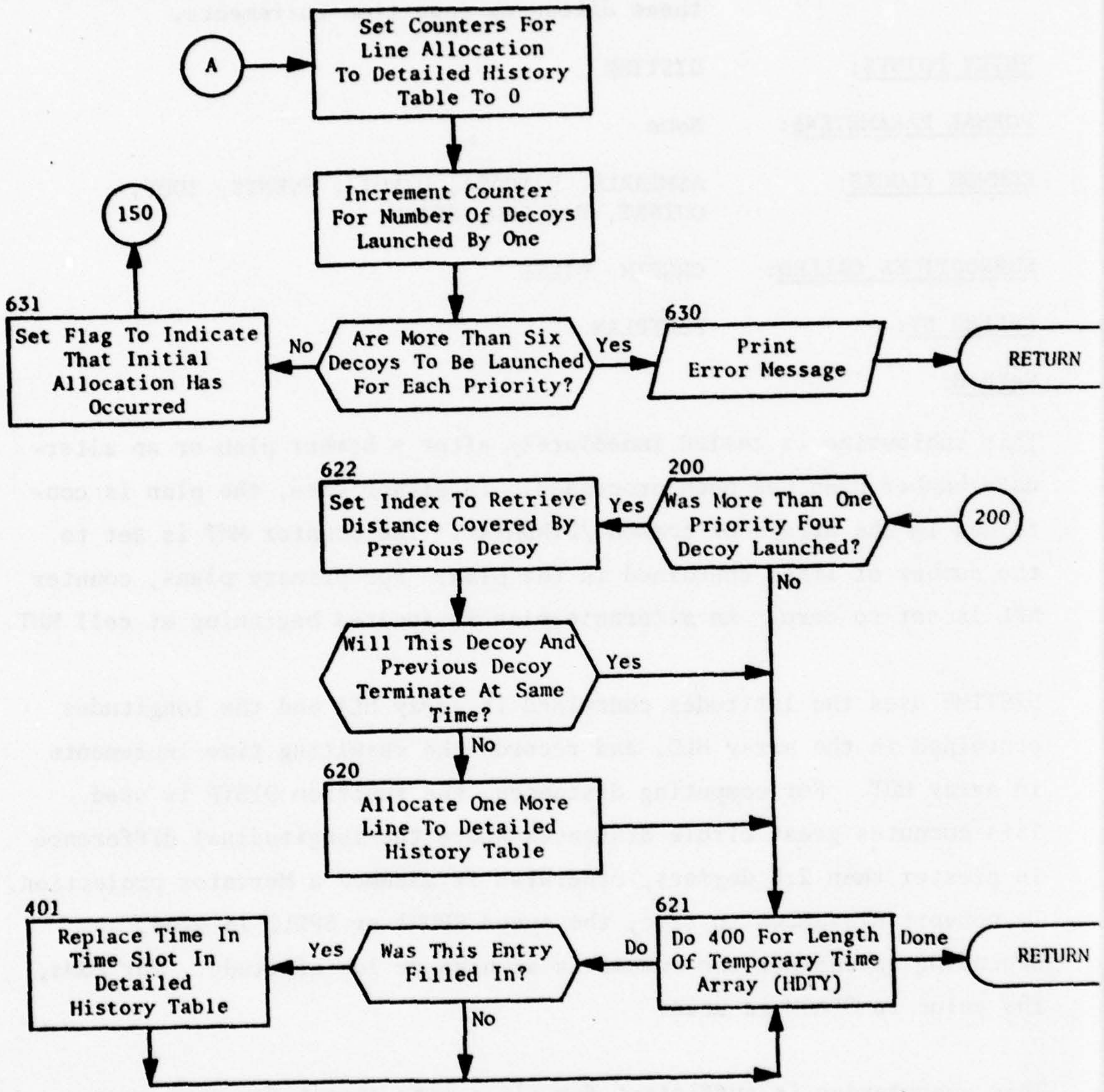


Figure 102. (Part 7 of 7)

.6.2.8 Subroutine DISTIME

PURPOSE:	To compute distances between events and convert these distances into time increments.
ENTRY POINTS:	DISTIME
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ASMTABLE, DINDATA, DINDT2, EVENTS, IOUT, OUTSRT, PAYLOAD, SPASM
SUBROUTINES CALLED:	CHGTIM, DISTF
CALLED BY:	PLNTPLAN

Method:

This subroutine is called immediately after a bomber plan or an alternate bomber plan has been processed. In either case, the plan is contained in the arrays of common /DINDATA/. The counter MHT is set to the number of lines contained in the plan. For primary plans, counter NPL is set to zero. An alternate plan is located beginning at cell MHT.

DISTIME uses the latitudes contained in array HLA and the longitudes contained in the array HLO, and records the resulting time increments in array HDT. For computing distances, the function DISTF is used. This computes great circle distances where the longitudinal difference is greater than 2.8 degrees, otherwise it assumes a Mercator projection. To convert distances to time, the speed SPDHI or SPDLO is used, depending on whether the bomber is at high or low altitude. For ASMs, the value in SPASM is used.

This computation is sufficient for all events except for zone crossings. This is because zone crossings are located or determined only

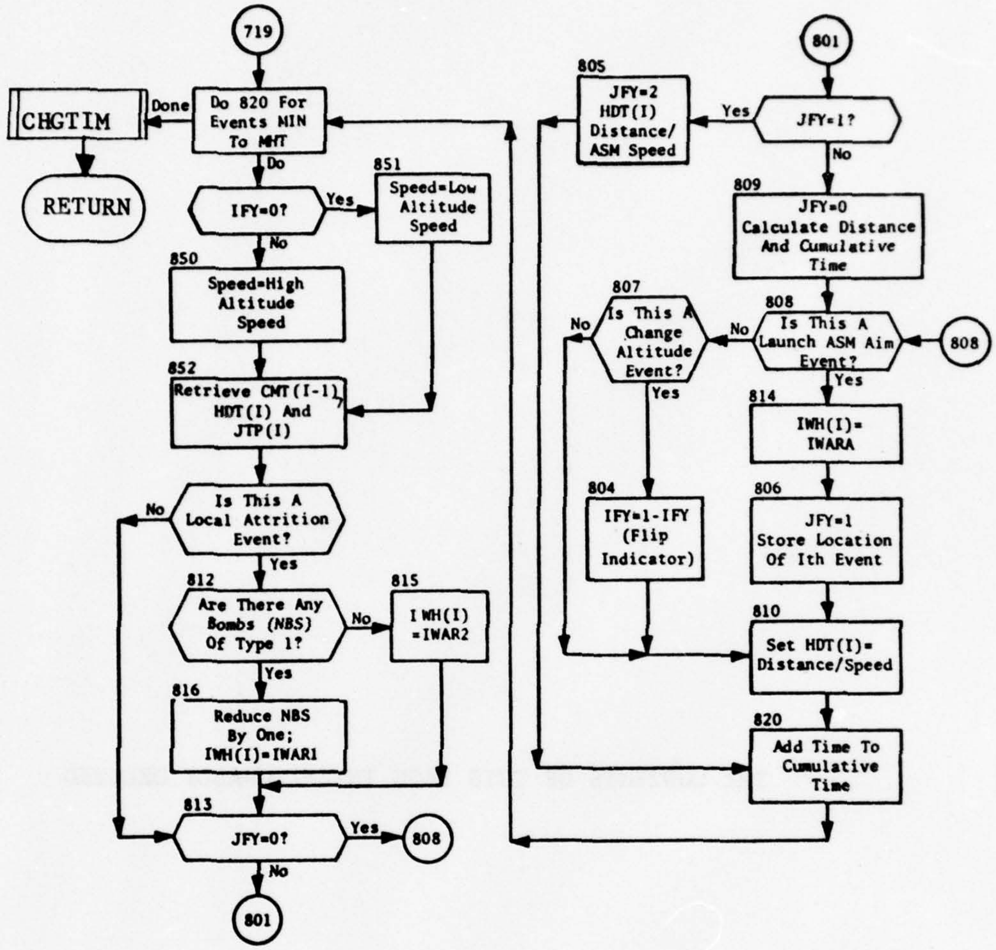


Figure 104. (Part 2 of 2)

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

THE CONTENTS OF THESE PAGES INTENTIONALLY DELETED

4.6.?.10 Subroutine FLYPOINT

PURPOSE: FLYPOINT is an integral part of block 40 in PLAN which adjusts events for ASM launches.

ENTRY POINTS: FLYPOINT, PREFL1, PREFL2, POSTFLY

FORMAL PARAMETERS: None

COMMON BLOCKS: ASMARRAY, LASM, OUTSRT

SUBROUTINES CALLED: DISTF, LAUNCH

CALLED BY: PLAN

Method:

PREFL1 determines the distance between the ASM target and the previous flypoint which was not an AIM ASM event.

PREFL2 calculates the distance between the ASM target and the previous flypoint.

POSTFLY finds the next flypoint, and calls subroutine LAUNCH to compute the ASM launch point.

Subroutine FLYPOINT is illustrated in figure 106.

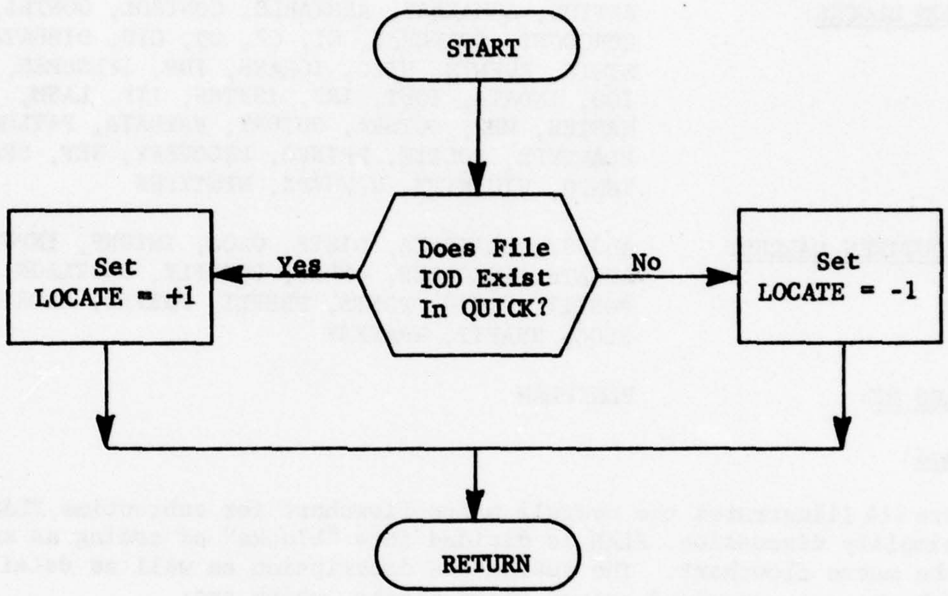


Figure 113. Subroutine LOCATE

4.6.?.15 Subroutine PLAN

PURPOSE: Subroutine PLAN develops detailed bomber sorties.

ENTRY POINTS: PLAN

FORMAL PARAMETERS: None

COMMON BLOCKS: ARTIME, ASMARRAY, ASMTABLE, CONTROL, CONTR1, CORCOUNT, CORRCHAR, C1, C7, C9, C10, DINDATA, DISTC, EVENTS, HILO, ICLASS, IDP, IFLGDPEN, IGO, INDATA, IOUT, IRF, ISRTNS, ITP, LASM, MASTER, MH2, OUTSRA, OUTSRT, PAYDATA, PAYLOAD, PLANTYPE, POLITE, PPINFO, RECOVERY, REF, SPASM, TEMPO, VICINITY, WPNGRPX, WPNTYPEX

SUBROUTINES CALLED: ADJUST, CLINDATA, DISTF, GLOG, INTERP, INPUT, LOCATE, LREORDER, ORDER, POSTFLY, POSTLAUN, POST17, POST4, POST8, PREFL1, PREFL2, REORDER, SLOG, SNAPIT, WRARRAY

CALLED BY: PLNTPLAN

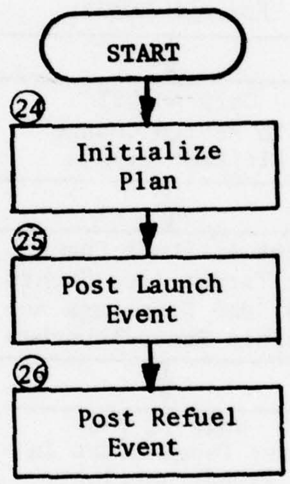
Method:

Figure 114 illustrates the overall macro flowchart for subroutine PLAN. To simplify discussion, PLAN is divided into "blocks" of coding as noted in the macro flowchart. The subroutine description as well as detailed flowcharts are organized around these blocks, which are:

BLOCK 20 - Determine type of plan
24 - Initialize plan
25 - Post Launch event
26 - Post Refuel events
27 - Initialize plan with respect to GOLOW range
30 - Process precorridor legs and apply GOLOW-1
31 - Post corridor events
40 - Adjust /OUTSRT/ for ASM events
50 - Apply GOLOW-2 before first target
60 - Post depenetration events

and are described in the following.

Block 20: Determine Type of Plan (figure 115)



*Circled numbers refer to start of coding blocks rather than to statement numbers.

Figure 114. Subroutine PLAN (Macro Flowchart)
(Part 1 of 2)

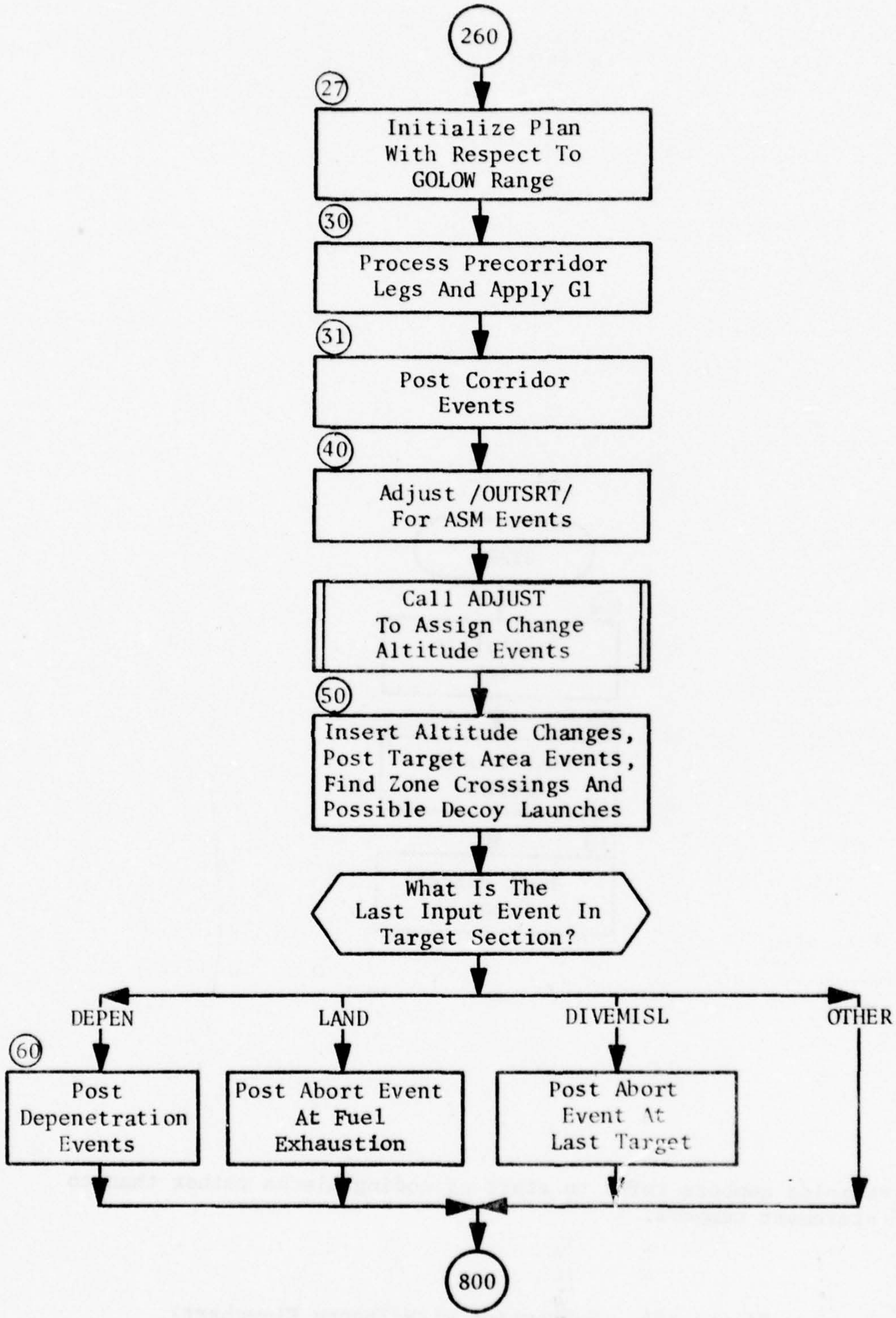


Figure 114. (Part 2 of 2)

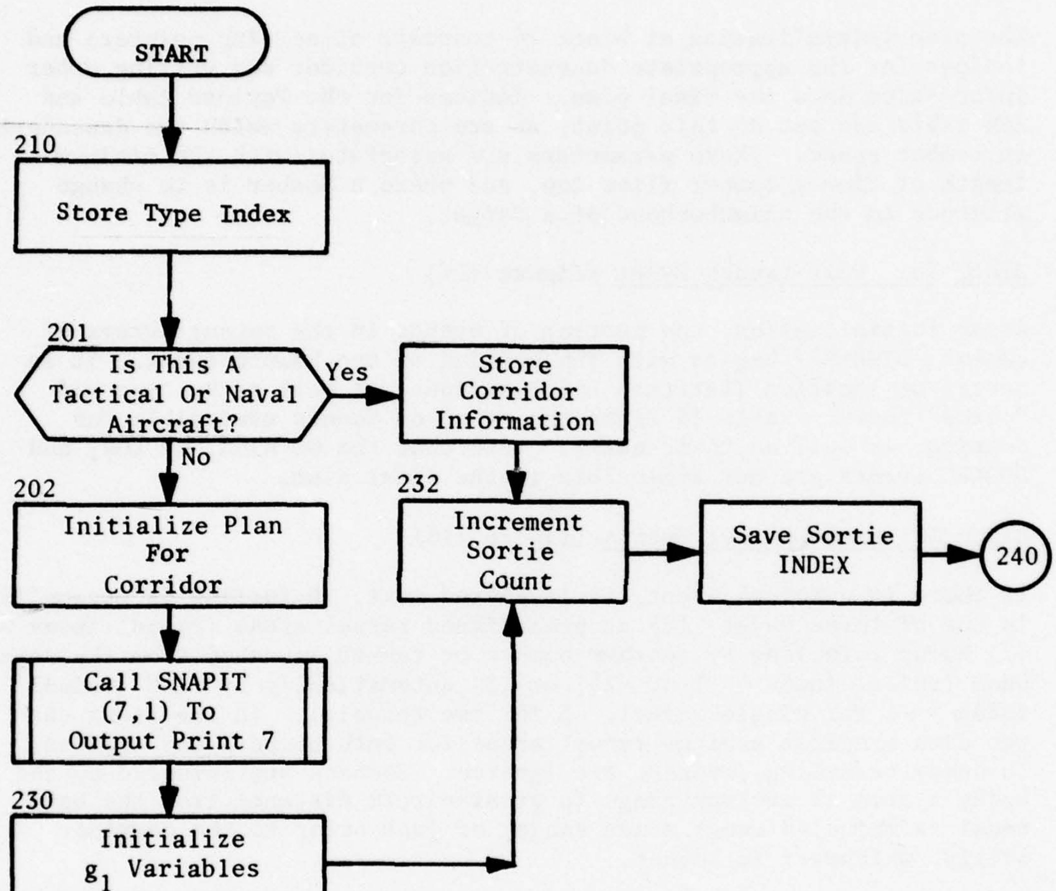


Figure 115. Subroutine PLAN
Block 20: Determine Type of Plan

SNAPIT is called to output Print 7 and the low altitude variable G1 is set.

Block 24: Initialize Plan (figure 116)

The plan initialization at block 24 consists of setting pointers and indices for the appropriate depenetration corridor and writing other information into the final plan. Indices for the Payload table and ASM table are set at this point, as are parameters which are dependent on bomber speed. These parameters are associated with the minimum length of time a bomber flies low, and where a bomber is to change altitude in the neighborhood of a target.

Block 25: Post Launch Event (figure 117)

After initialization, the posting of events in the output arrays of common /DINDATA/ begins with the posting of the Launch event. It is posted by location (latitude and longitude) as well as by type and "place" index. Table 16 lists the types of events admissible for posting, as well as their names. Note that the GO HIGH, GO LOW, and DOGLEG events are not admissible in the final plan.

Block 26: Post Refuel Events (figure 118)

If there is a Refuel event, it is posted next. Refueling is accomplished in one of three ways: (1) at preassigned refuel areas (refuel index ≥0), (2) buddy refueling by another bomber or tanker launched from the same base (refuel index = -1 or -2), or (3) automatically by PLAN (refuel index = -4 for single refuel, -5 for two refuels). In the first case the data preparer assigns refuel areas for both bombers and tankers. In buddy refueling, tankers are ignored. Bombers are refueled by the buddy system at maximum range (a great-circle distance from the base equal to refueled range minus range) or just prior to the corridor origin, whichever is sooner.

When a bomber is to be assigned a refuel area by PLNTPLAN the buddy refuel point, X, is first computed a distance ΔR from the base on a great circle between it and the corridor entry, as for buddy refueling. See figure 119. ΔR is the difference between refueled range and range. If there already exist refuel areas which are within ΔR of the base and within some specified distance, D, of the point X, the area nearest X is assigned as the refuel point. Otherwise the point X is assigned and is added to the list of refuel areas.

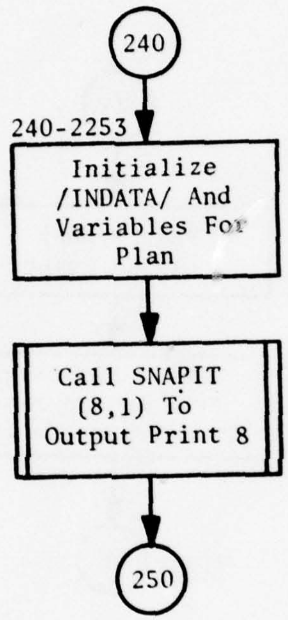


Figure 116. **Subroutine PLAN**
Block 24: Initialize Plan

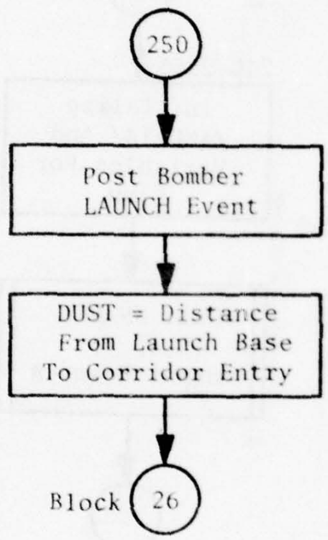


Figure 117. Subroutine PLAN
Block 25: Post Launch Event

point is also noted. If attrition begins at a point, this is noted by entering a 1, 2, or 3 in array JAPTYPE, depending upon whether this is the first, second, or third section. Similarly, if attrition ends at the point, the number 4, 5, or 6 is entered. Thus point 1 is labeled with a 1, and point 3 with a 4, to indicate the beginning and end of the first attrition section. This example, of course, describes an extreme situation where attrition occurs in three separate sections. Usually, there will be attrition in at most one section. The program must know which doglegs have attrition in order to know where to apply the low altitude range G_1. In the figure example, suppose G_1 = 180 miles, then 112 miles would be applied against the first dogleg back of the corridor origin, 38 miles applied to the second dogleg. The balance of 30 miles would be applied to the 5th dogleg beginning with point 5 and ending midway between points 5 and 6. As a result, GO LOW and GO HIGH events would be posted as indicated in the figure. The posting of a GO HIGH at the corridor origin depends on the value of G_2.

This section also sets up arrays which contain the event numbers of all those precorridor events which might possibly call for the launching of a decoy. These arrays are called LOHIMHT and LDMHT. The event number is stored in LOHIMHT for events of priority 2. (See subroutine DECOYADD for table of priorities.) The event number is stored in LDMHT for a launch of priority 3 (or, after the first such event, priority 5). For the priority 5 launches, the distance to be covered by the decoy is accumulated in a corresponding word of array DELDIS.

The flag JA is set to 1 when the beginning of an attrition section is encountered, and back to 0 at the end. IDEL is the indicator to be compared against JAPTYPE in order to determine the beginning and end of attrition events. The counter JDO is advanced each time an additional Change Altitude event is added.

<u>Block 31: Post Corridor Events</u> (figure 123)

G_1 is measured out and the necessary change altitude events determined. The possible Decoy Launches then are posted by filling array LMHT with the event number (from array LDMHT or LOHIMHT) and by filling array

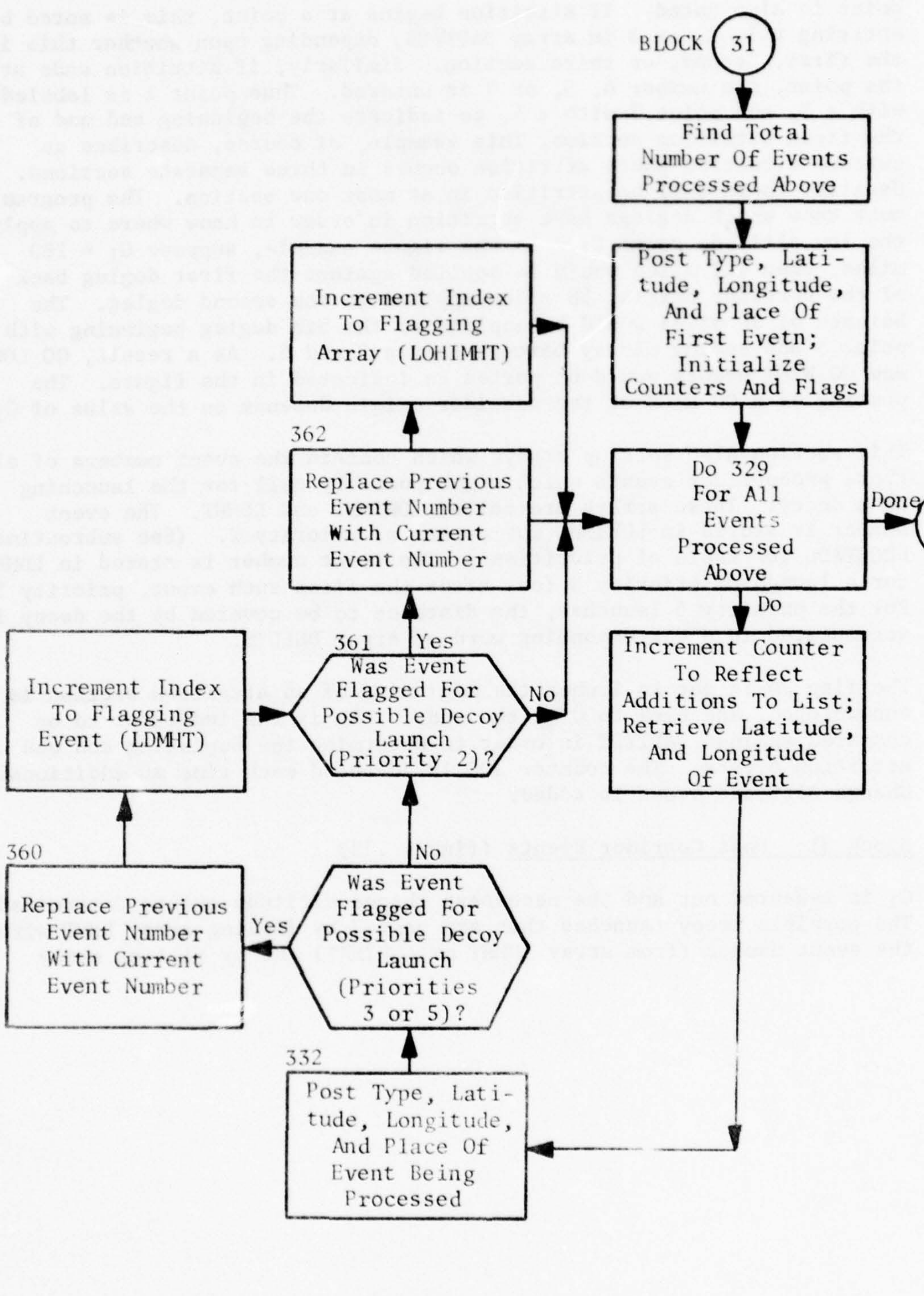


Figure 123. Subroutine PLAN
Block 31: Post Corridor Events
(Part 1 of 4)

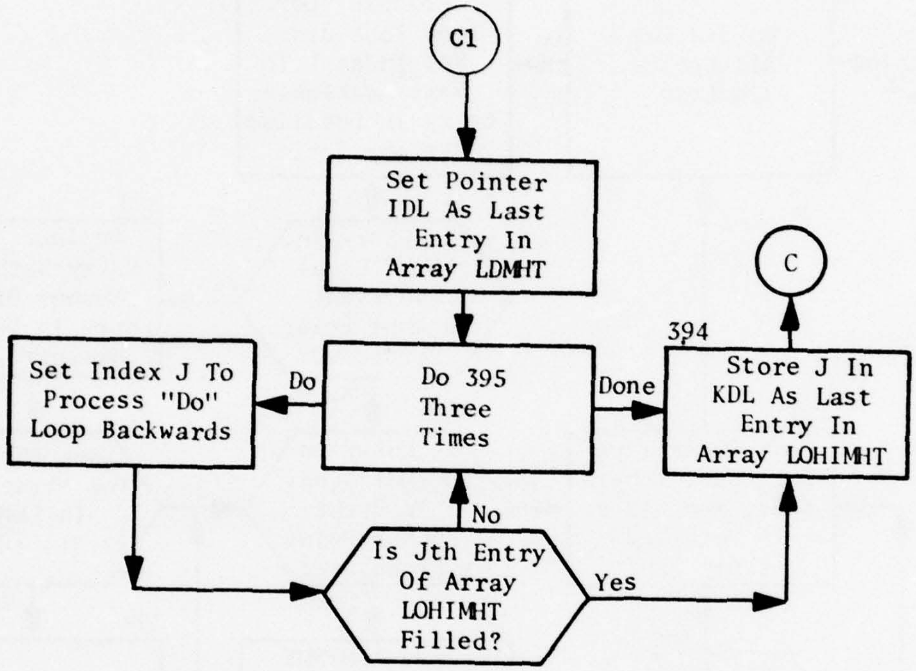


Figure 123. (Part 2 of 4)

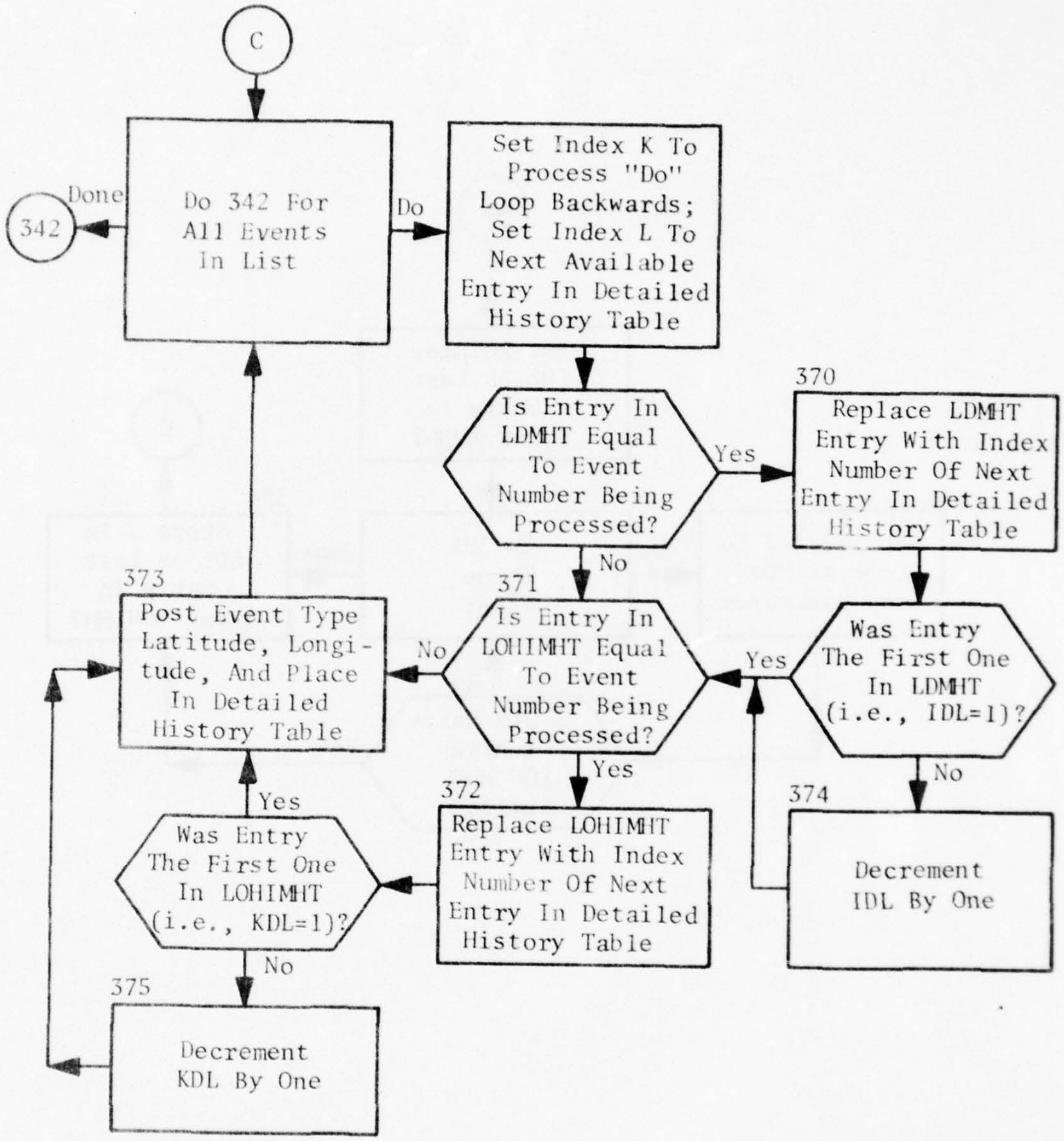


Figure 123. (Part 3 of 4)

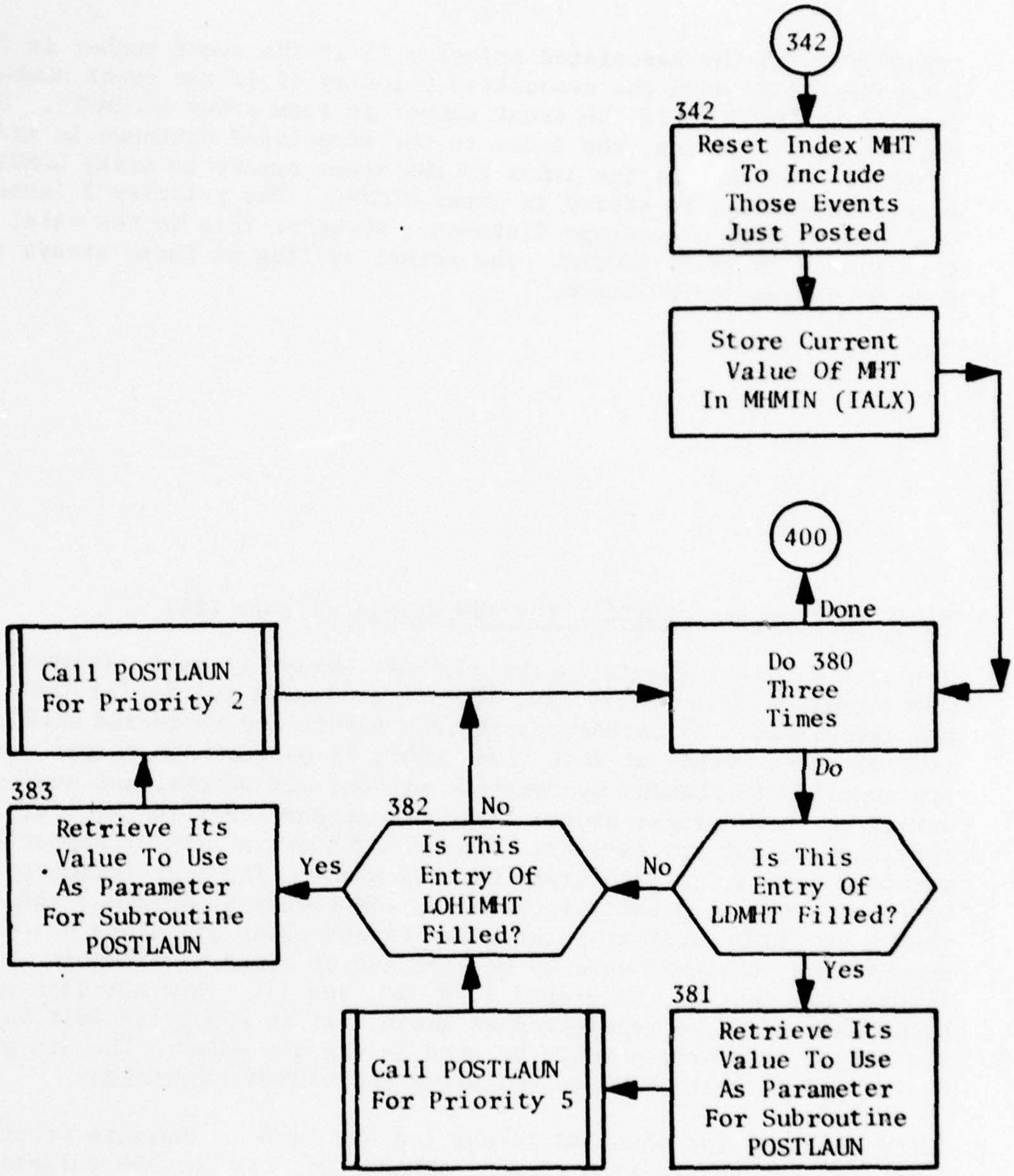


Figure 123. (Part 4 of 4)

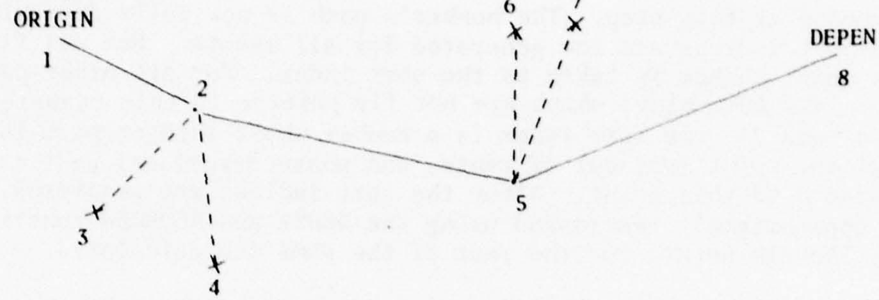
LPRIORITY with the associated priority (5 if the event number is from array LPRIORITY with the associated priority (5 if the event number is from array LDMHT, 2 if the event number is from array LOHIMHT). For the priority 5 launch, the index to the associated distance in array DELDIS is the same as the index to the event number in array LDMHT; hence, this index is stored in array NDCYRQ. The priority 2 launch does not require a coverage distance. Whenever this is the case, a 1 is stored in array NDCYRQ. The actual filling of these arrays is done by subroutine POSTLAUN.

Block 40: Adjust /OUTSRT/ for ASM Events (figure 124)

The list of input events in the /OUTSRT/ arrays is next examined for ASM events. If there are any, the aim or launch points for them are now calculated. If necessary, the ASM events are reordered within the list of other events at this time. This is because ASM target events are supplied to PLANOUT by POSTALOC without aim points, and approximately in their proper order. They may appear later in the lists than they should, but not earlier. Figure 125 shows a list of happenings with ASM events, to illustrate what is meant. The list indicates a DROPBOMB event at point 2 followed by ASM events at points 3 and 4, then a DROPBOMB event at point 5 and an ASM event at points 6 and 7. Suppose that the ASMs were to be launched or aimed as shown in figure 125; that is, at points 1, 9, 10, and 11. Then the list of happenings would be rearranged as shown. If an ASM point fell before the origin, the origin would be used as the aim point. The aim points 9, 10, and 11 would be computed using the LAUNCH subroutine.

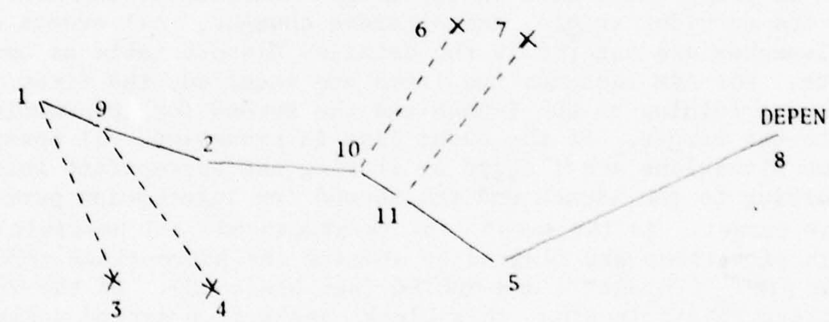
The processing for block 40 is carried out in four separate steps, utilizing the arrays from common /ASMARRAY/. (1) The ASM targets are first examined to see if they are in range of the origin or some other prior fly point such as a drop bomb point. Those which are not are flagged by setting the corresponding cell of array IFLY to 1. In the example in figure 125, points 3 and 4 would not be flagged, but points 6 and 7 would. (2) The ASM targets flagged in the last step would again be examined in the second pass to see if any were in range of a previous ASM fly point. In the example shown, point 7 would be in range of ASM target 6. The aim point (No. 10) for ASM target 6 is

(a) Before Adjustment



Point	IBTYPE
1	DOGLEG
2	DROPBOMB
3	AIMASM
4	AIMASM
5	DROPBOMB
6	AIMASM
7	AIMASM
8	DEPEN

(b) After Adjustment



Point	IBTYPE
1	DOGLEG
3	AIMASM
4	AIMASM
2	DROPBOMB
6	AIMASM
7	AIMASM
5	DROPBOMB
8	DEPEN

Figure 125. Illustration of ASM Event Adjustment

computed at this step. The bomber's path is now fully determined. (3) Sort indices are now generated for all events. For all fly points, the point number is taken as the sort index. For all other points, (i.e., all ASM points which are not fly points; in this example, points 3, 4, and 7), the sort index is a number whose integer part is the earliest point just out of range, and whose fractional part is the distance to this point. After the sort indices are generated, the list is appropriately rearranged using the ORDER and REORDER functions. (4) The aim points for the rest of the ASMs are calculated.

Subroutine FLYPOINT, which has the entries PREFL1, PREFL2, and POSTFLY, is logically an integral part of this coding block. Subroutine LAUNCH is called only by POSTFLY.

If the last event is an ASM event, the depenetration corridor is re-selected.

The locations of appropriate Change Altitude events associated with the ranges G_2 and G_3 are now calculated by subroutine ADJUST. If the target area was found to be degenerate, blocks 50 and 60 are skipped.

Block 50: Apply GOLOW-2 Before First Target (figure 126)

Block 50 posts all events in the target area, including ASM launches from the corridor origin, and altitude changes. All events except ASM launches are entered in the detailed History table as one-line events. For ASM launches two lines are required, the first for information pertaining to the launch and the second for information pertaining to the target. As the event list is processed, all possible Decoy Launch situations are flagged by storing the appropriate information pertaining to the launch and the second for information pertaining to the target. As the event list is processed, all possible Decoy Launch situations are flagged by storing the appropriate information in arrays LMHT, LPRIORITY, and NDCYRQ (see block 30). If the value of attribute PAYALT is HIGH, this block checks if a second segment of low-altitude flight is planned after the last target. If so, an altitude change event is placed after the last target.

Processing is terminated with the occurrence of the input events DEPEN, LAND, or DIVEMISL. For DEPEN, a normal exit is made to block 60 described in the next section. For LAND, post a GO HIGH event if currently at low altitude and determine an abort point on the line connecting the last target and the depenetration point at which the bomber's range (fuel) is exhausted. If the bomber would normally use all available range in reaching its last assigned target, a hypothetical abort point is established 5 minutes' flight distance beyond the target. For DIVEMISL, an abort event is posted immediately at the last target.

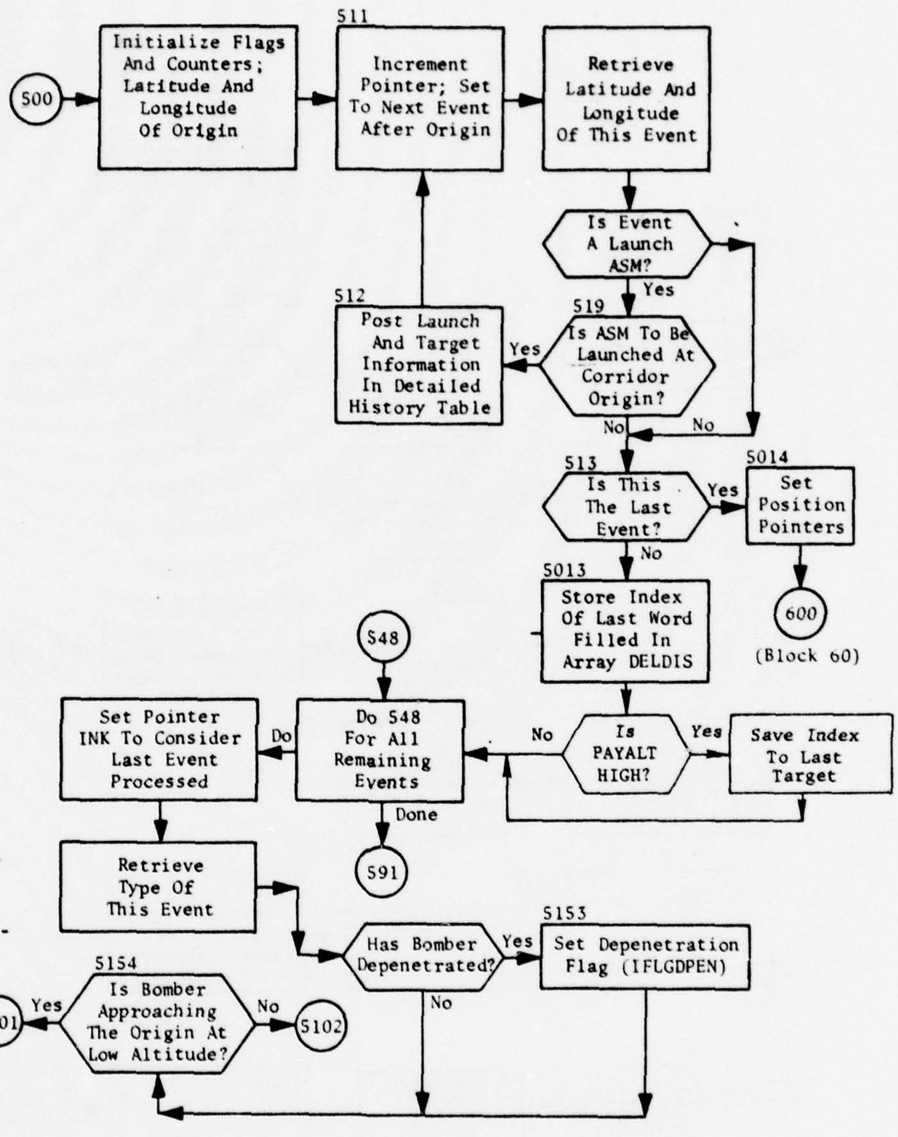


Figure 126. Subroutine PLAN
Block 50: Apply GOLOW2 Before First Target
(Part 1 of 5)

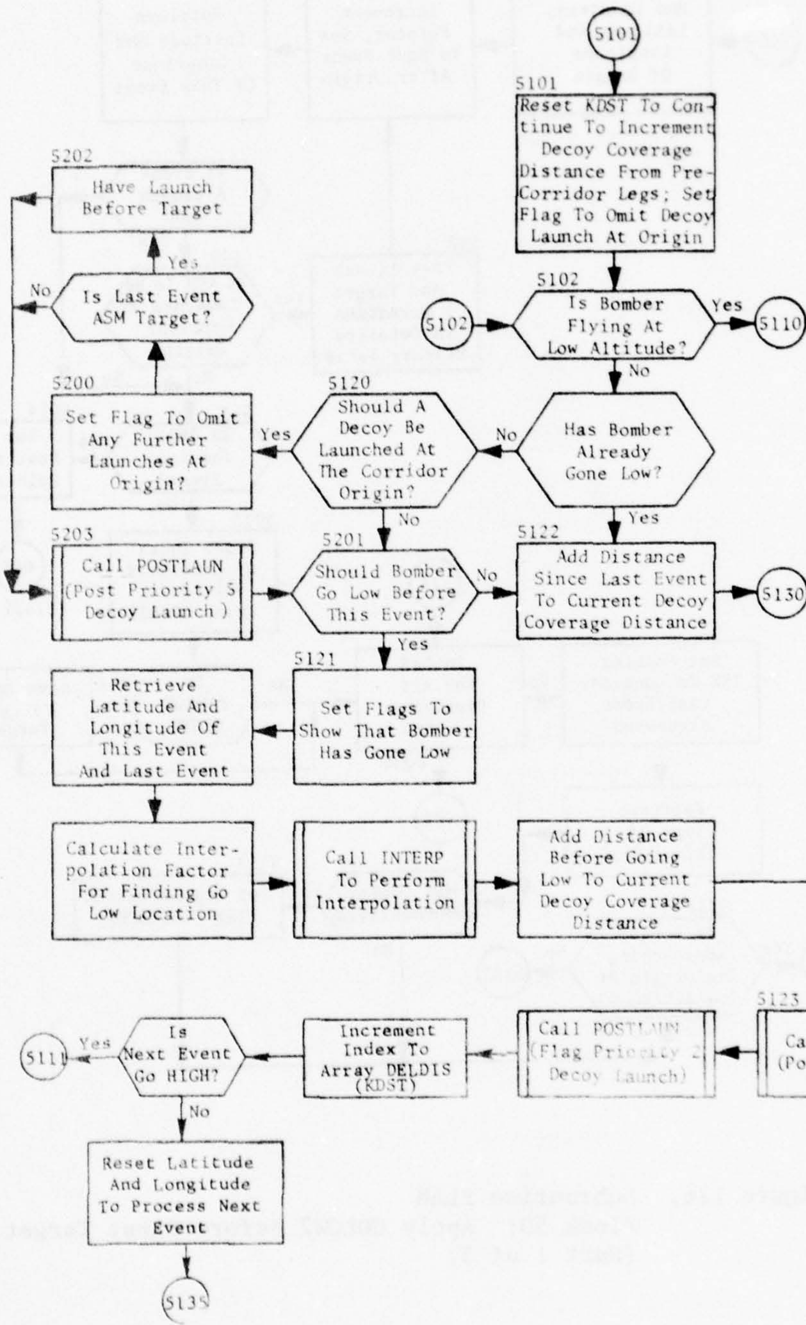


Figure 126. (Part 2 of 5)

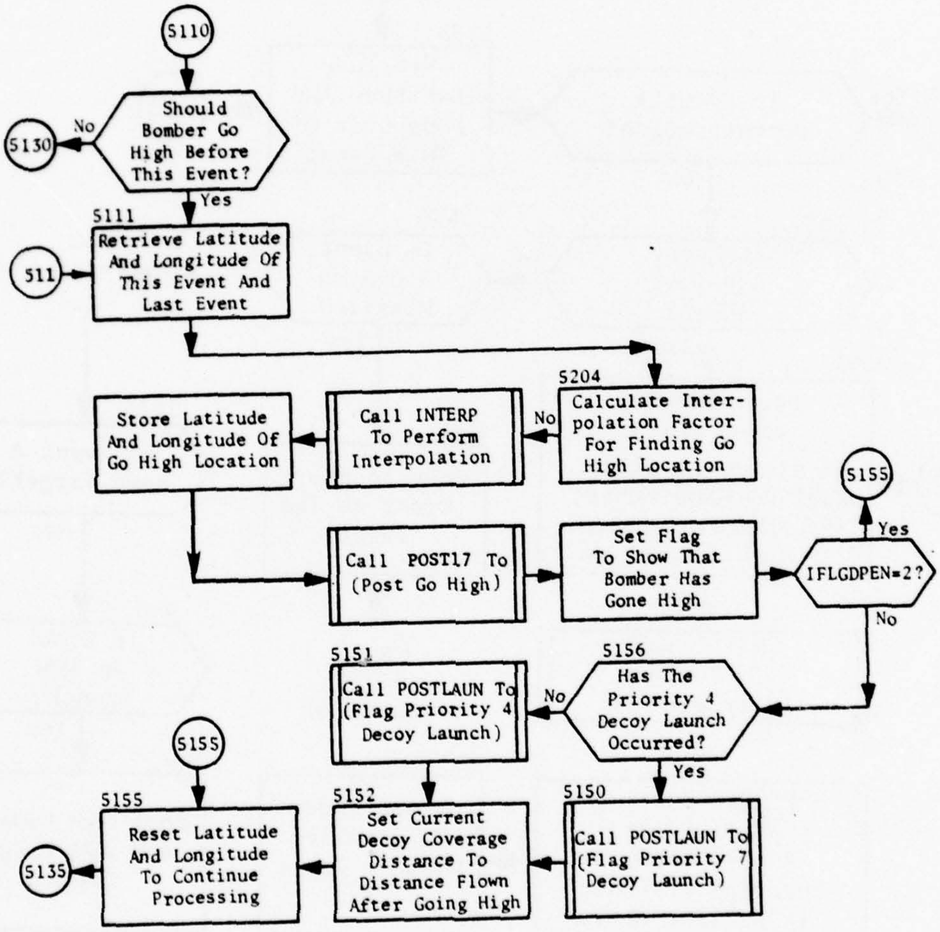


Figure 126. (Part 3 of 5)

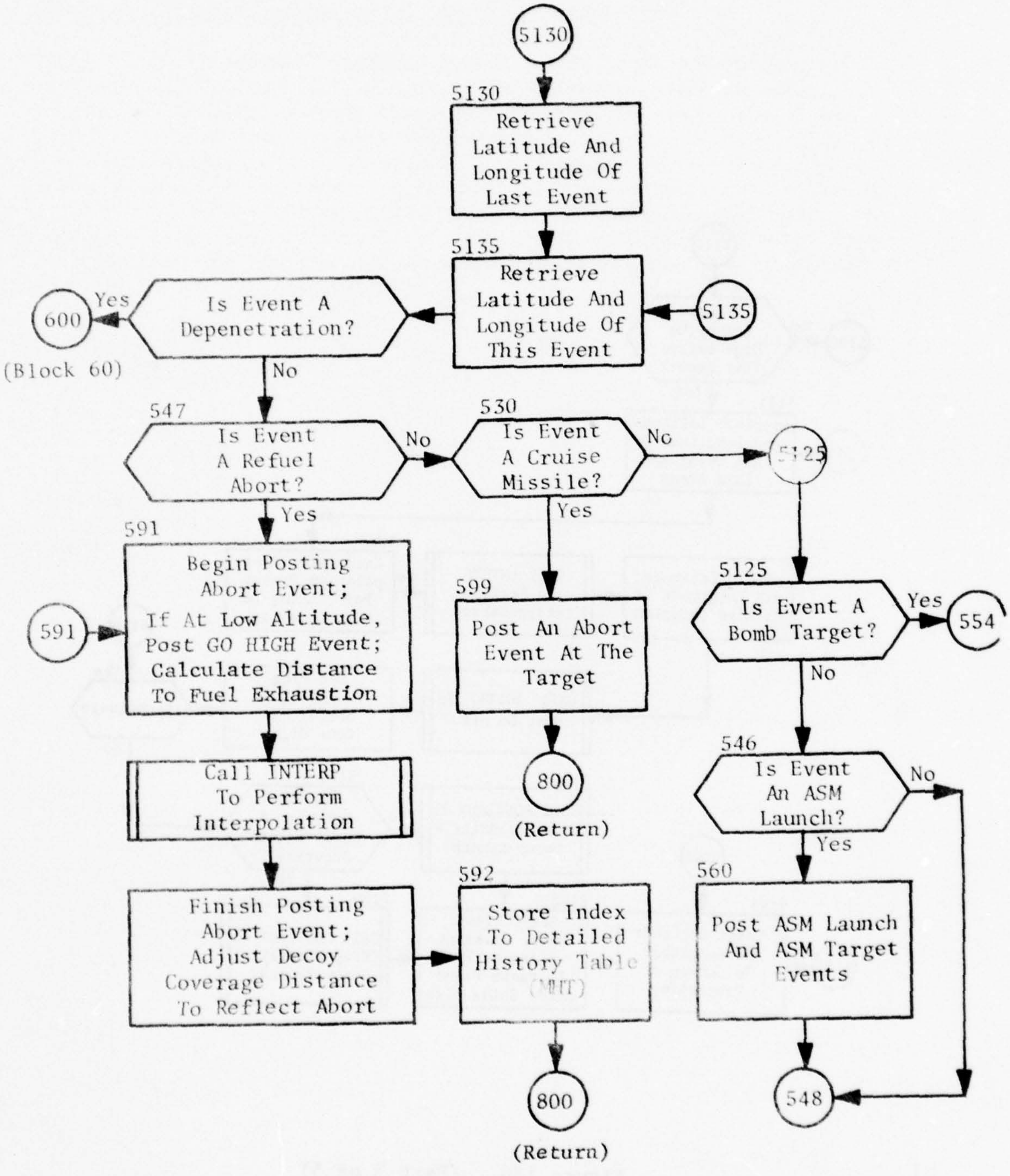


Figure 126. (Part 4 of 5)

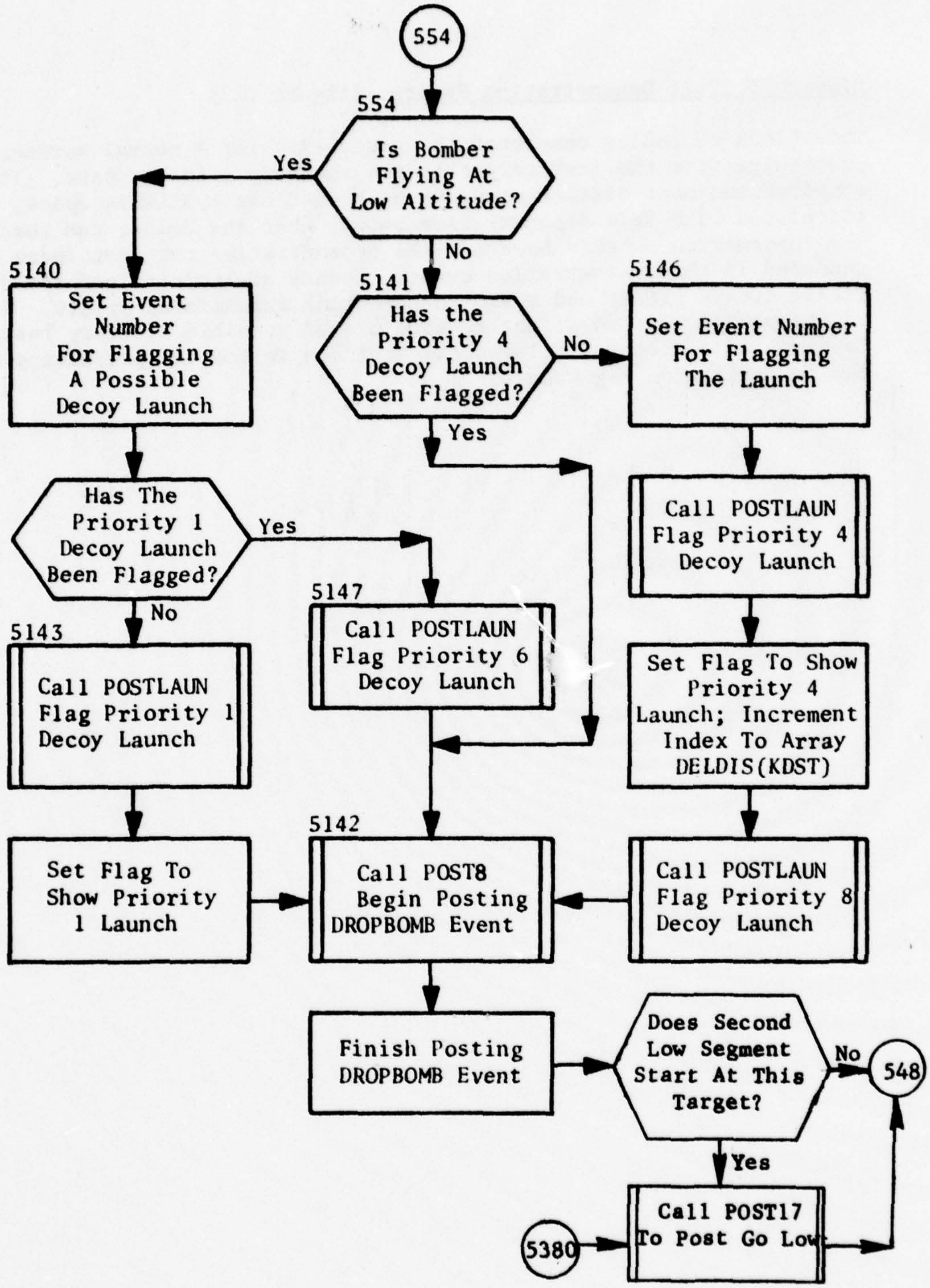


Figure 126. (Part 5 of 5)

Block 60: Post Depenetration Events (figure 127)

This block of coding completes the processing for a normal sortie, processing from the last target to the recovery point or base. It computes the most distant recovery base that has available space, associated with this depenetration point, that the bomber can reach. The information on this base and the depenetration corridor index are recorded in the depenetration event. Counts are updated and saved within arrays NAMCAP and NUSED for eventual summarizing prints. In addition, the time of flight to each of the possible recovery bases is computed and stored. These calculations follow the processing of the depenetration leg events.

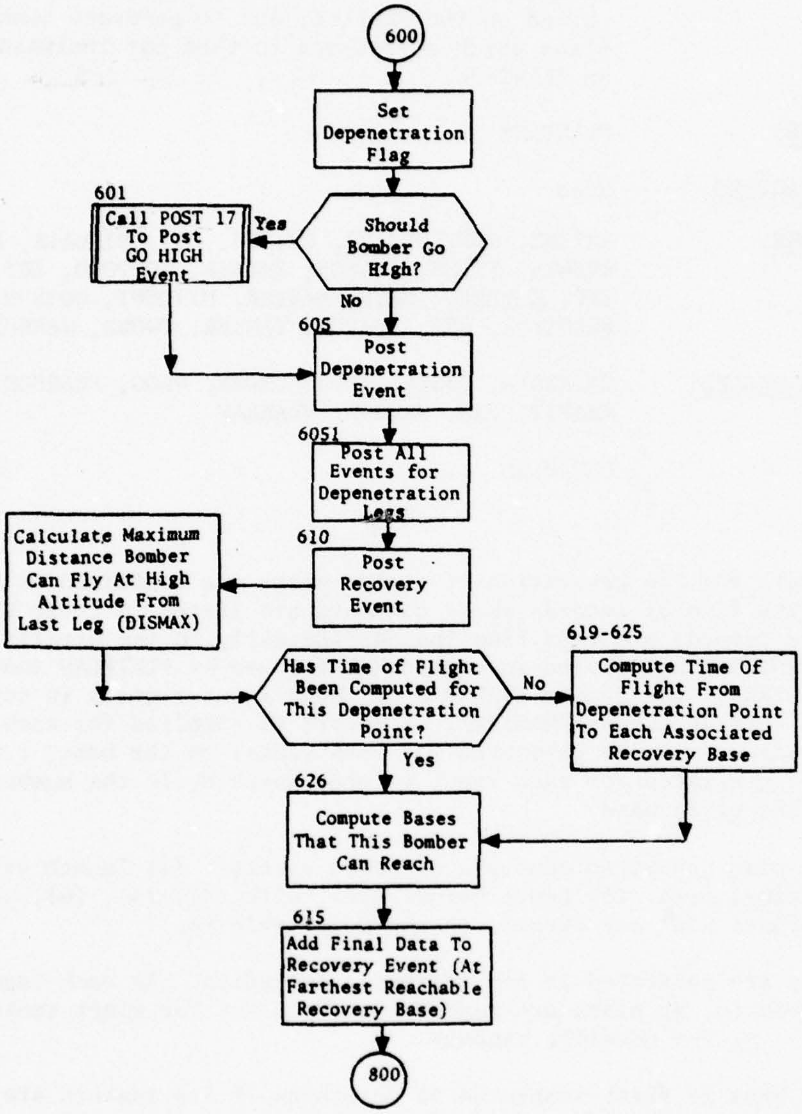


Figure 127. Subroutine PLAN
Block 60: Post Depenetration Events

4.6.2.16 Subroutine PLANTANK

PURPOSE: To process the tanker records originally contained on the BASFILE, and to generate tanker plans which correspond to them for inclusion on PLANTAPE.

ENTRY POINTS: PLANTANK

FORMAL PARAMETERS: None

COMMON BLOCKS: ARTIME, CONTROL, C7, C8, C9, C10, DINDATA, DINDT2, EVENTS, FILES, ICLASS, INDATA, IOUTOLD, IRF, IRFTK, ITP, KEYLENG, KEYS, MASTER, MYIDENT, OUTSRT, PLANTYPE, REF, SNAPON, TANKER, TWORD, WAROUT

SUBROUTINES CALLED: CLINDATA, DISTF, IPUT, ORDER, SLOG, SNAPCON, SNAPIT, VAM, WRWORD, WRARRAY

CALLED BY: PLNTPLAN

Method:

The input data for the generation of tanker plans are contained on the BASFILE in the form of records whose contents are listed in table 38. All of these records are read from the BASFILE early in the program by subroutine INITANK and stored in common /C7/ for use by PLNTPLAN and subroutine PLANTANK. The number NTANKBAS of these input records is supplied to PLNTPLAN through common /MASTER/. A record is supplied for each tanker base. A separate plan is generated for each tanker on the base; i.e., N_t plans are generated for each input record, where N_t is the number of tankers on the given base.

Each tanker plan generated consists of seven events: (1) Launch event, (2) Enter Refuel Area, (3) Leave Refuel Area, with (4), (5), (6), and (7) as alternate Recovery events, as shown in table 39.

Tanker plans are generated in the following operation. As each input record is read in, N_t plans are generated: N_a plans for alert tankers, and then N_t - N_a for nonalert tankers.

Each tanker base is first inspected to determine if its tankers are to be automatically allocated. Then, after all bases have been inspected, PLANTANK fills common block/C9/ with required data and calls subroutine VAM to allocate those tankers to specific refuel areas in such a way as to minimize the total miles flown by them while servicing all bomber requests.

When VAM has returned its solution, PLANTANK allocates any extra tankers and then proceeds to calculate the time schedules for individual flights.

In the second-strike case, all tankers are sent to their assigned refuel areas at the earliest possible moment, considering delays before launch due to alert or nonalert status as well as the travel time required between base and refuel area.

In the first-strike case, however, they are scheduled as follows. Bombers have been scheduled by PLNTPLAN to arrive at specific refuel areas over a period of time (which may be several hours) so as to satisfy the requirements associated with the CORBOMB input parameter. These bomber refuels have been posted in the matrix IARVLS/ARVLS (I,J) where J indicates data for the Jth refuel to be scheduled by PLNTPLAN, I=1 contains the scheduled time of the Jth refuel, I=2 contains the assigned refuel area.

As each tanker is processed, the IARVLS array is searched for the first unserviced bomber refuel which is to occur at the refuel area to which the tanker has been assigned by subroutine VAM. When found, the bomber time of arrival is retrieved, the tanker is scheduled to launch so as to arrive at the refuel area .1 hour prior to the bomber, and the IARVLS entry is set to zero to indicate that the bomber has been serviced. If the search finds no unserviced bomber at the refuel area, the tanker is extra, thus PLANTANK schedules it to arrive .1 hour before the earliest bomber at the area (stored in array ARTIME).

After scheduling has beem completed, distances from refuel area to recovery bases are calculated for each tanker, the recovery events are ordered by ascending distance, and PLANTAPE, and printed reports are output with tanker plans according to user options.

Figure 128 illustrates subroutine PLANTANK.

Table 38. Tanker Input Record

ITEM	Fortan Name	Symbolic Name
Tanker base index	INDEXTK	
Base latitude	TKLAT	
Base longitude	TKLONG	
Refuel area	IREFTK	
Number of tankers per squadron	NPSQNTK	N_t
Number of tankers on alert per squadron	NALRTK	N_a
Tanker speed	SPEEDTK	V_t
Alert delay	DLYALTK	D_a
Nonalert delay	DLYALTK	D_n
Total time on station	TTOS	
Tanker type	ITYPETK	
Tanker range	TANKRNGE	

Table 39. Tanker Plan

Event Type	Time	Place
Launch	Delay	INDEXTK
Enter Refuel Area	$DIST/V_t$	IREFTK as set by PLANTANK
Leave Refuel Area	TTOS	IREFTK as set by PLANTANK
$Recover_1$	DI_1/V	$(RCBLAT, RCBLONG)_1$
$Recover_2$	DI_2/V_t	$(RCBLAT, RCBLONG)_2$
$Recover_3$	DI_3/V_t	$(RCBLAT, RCBLONG)_3$
$Recover_4$	DI_4/V_t	$(RCBLAT, RCBLONG)_4$

Where DIST = Distance from tanker base to refuel area

DI_x = Distance from refuel area to recovery $base_x$

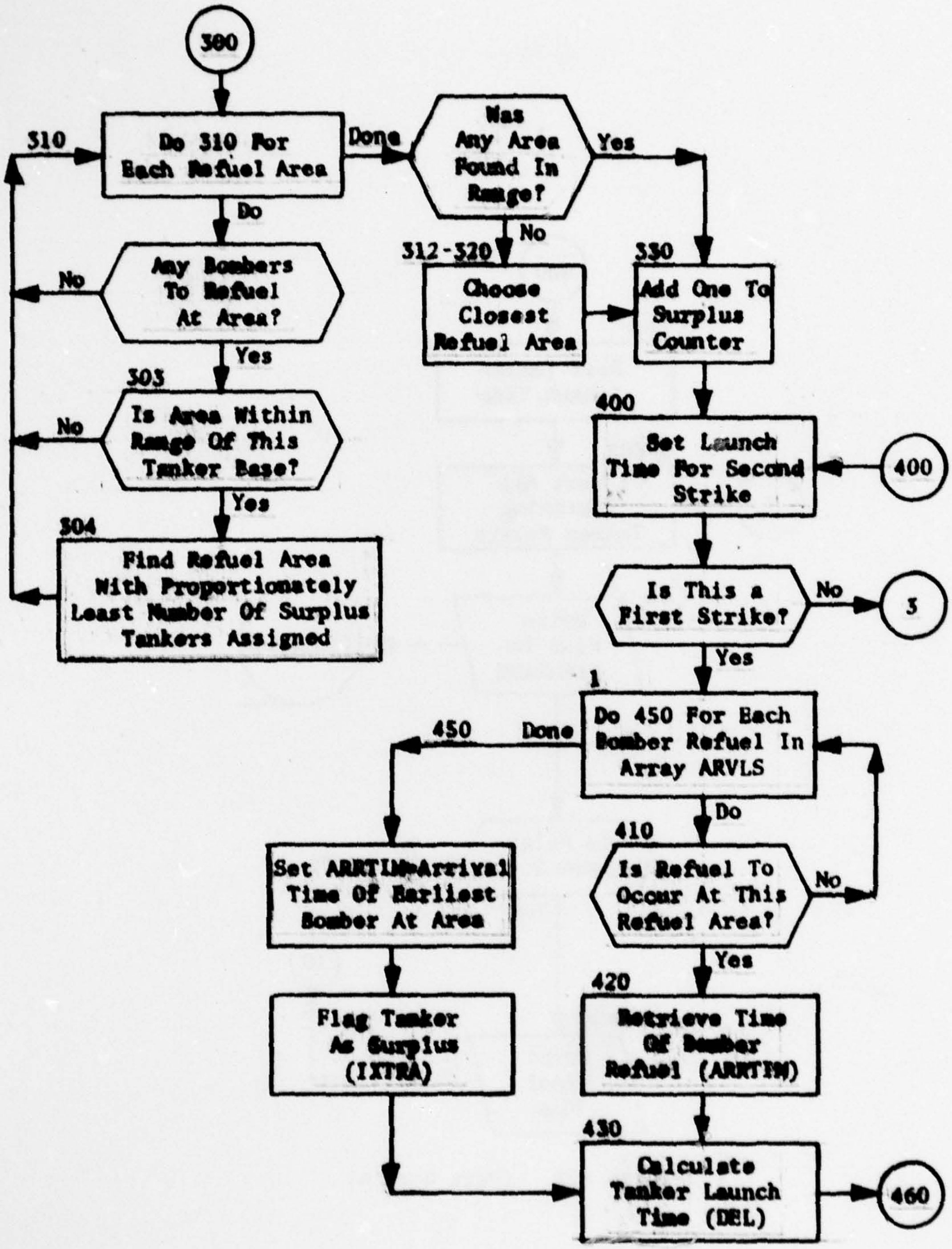


Figure 128. (Part 3 of 4)

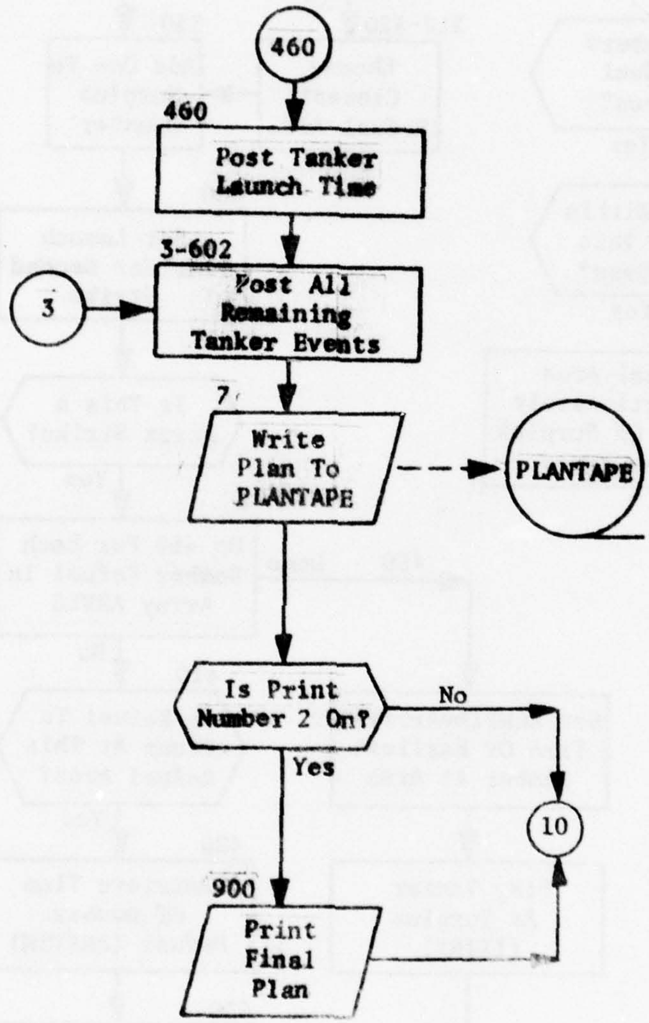


Figure 128. (Part 4 of 4)

4.6.2.17 Subroutine PLANTMIS

PURPOSE:	To control the processing of all missile plans input from the STRKFILE.
ENTRY POINTS:	PLANTMIS
FORMAL PARAMETERS:	None
COMMON BLOCKS:	BLOCK, CONTROL, CONTR1, FILES, ITP, KEYLENG, KEYM, MISCT, MRVFLG, NAVAL, OUTSRA, OUTSRT, PAYLOAD, SNAPON, TIMELINE, TWORD, WPNGRPX, DELTA, WAROUT, WPNTYPEX, MAX
SUBROUTINES CALLED:	ABORT, CHGCHK, GLOG, IGET, IPUT, KEYMAKE, RDARRAY, SNAPIT, TIMELNCH, TIMEME, WRARRAY, WRWORD
CALLED BY:	PLNTPLAN

Method:

This subroutine reads and processes missile plans from STRKFILE or STRKCHNG. It is called whenever PLNTPLAN reads a missile record.

The STRKFILE or STRKCHNG record is first moved from /OUTSRT/ to /BLOCK/. To facilitate processing, most of the data is then transferred to the TDATA array in common /TIMELINE/. Table 40 shows the variable placement in arrays BLOCK and TDATA.

The remainder of PLANTMIS merely outputs missile plans in the correct format onto the PLANTAPE. If all events have been deleted from a record that record is not output on the PLANTAPE. This situation is identified when target index is zero.

When PLANTMIS has completed processing a missile record, it reads the next plan from the STRKFILE and calls CHGCHK to read the STRKCHNG. If the information read is another missile record, PLANTMIS processes it without returning to PLNTPLAN; otherwise it returns.

Table 40. Arrays TDATA/ITDATA and BLOCK/LOCK Used in PLANTMIS and TIMELNCH

TDATA/ITDATA INDEX	ATTRIBUTE	BLOCK/LOCK INDEX
---	STRKFILE or EVENTAPE Record words 1-18	1-18
1-18	Missile indices	19-36
19-36	Site indices (from data base)	37-54
37-54	Target indices (from data base)	55-72
55-72	Offset latitude (DLAT)	73-90
73-90	Offset longitude (DLONG)	91-108
91-108	Flight Time (hours)	109-126
109-126	Weapon site latitude	127-144
127-144	Weapon site longitude	145-162
145-162	Target latitude	163-180
163-180	Target longitude	181-198
181-198	Target designator	199-216
199-216	Target task and owner	217-234
217-234	Target country	235-252
235-252	Target flag	253-270

PLAN01, the first overlay of PLANOUT, has the option of adding missile records to the STRKCHNG file whereby creating more records than on the STRKFILE. Parameter ICHANGE is set negative when these STRKCHNG records are to be processed.

Local array IHOB is used to store the weapon height of burst information. Logical array MMHOB contains the input HOB information from the LOCK array. This information is used to set the HOB code for each missile strike.

All processing of missile plans is done in subroutine PLANTMIS, TIMELNCH, and LNCHDATA.

Figure 129 illustrates subroutine PLANTMIS.

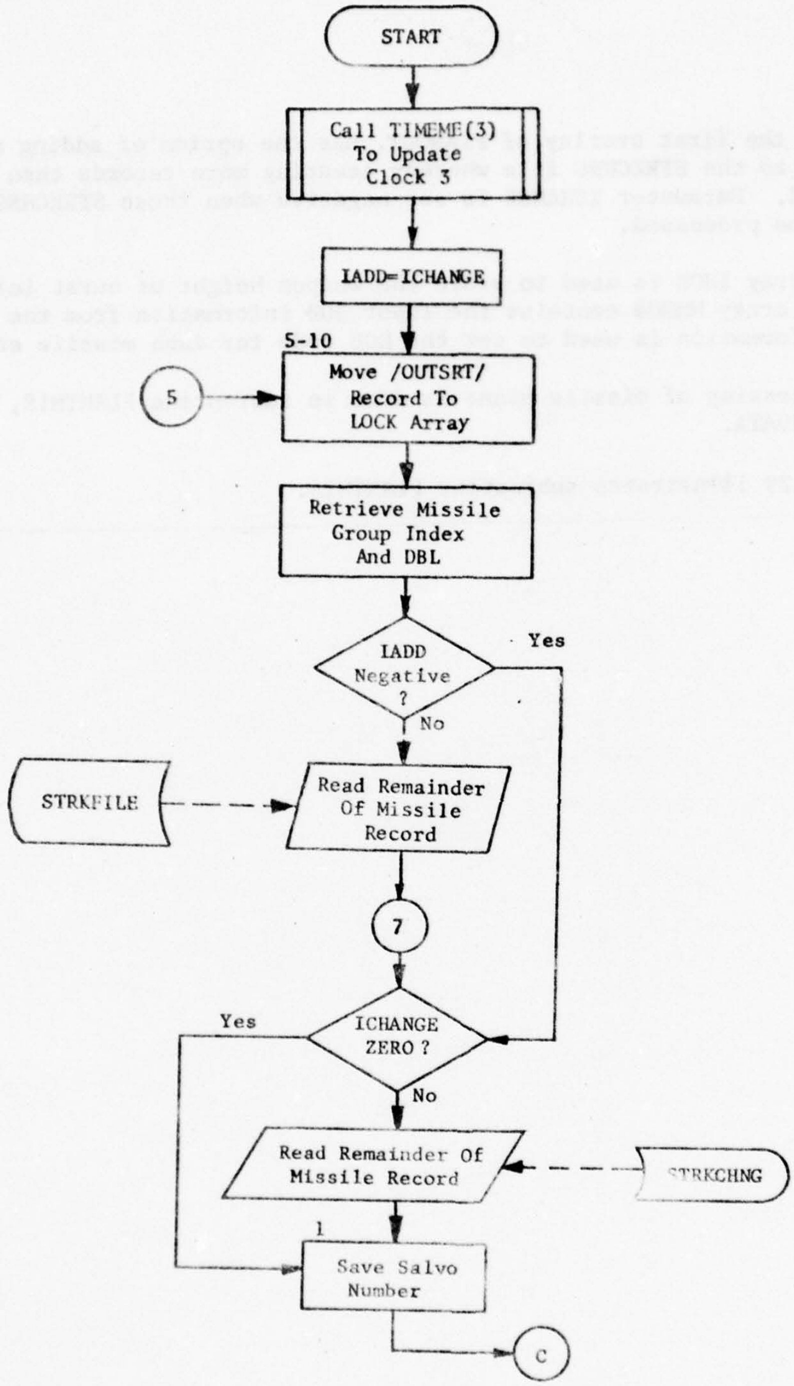


Figure 129. Subroutine PLANTMIS (Part 1 of 5)

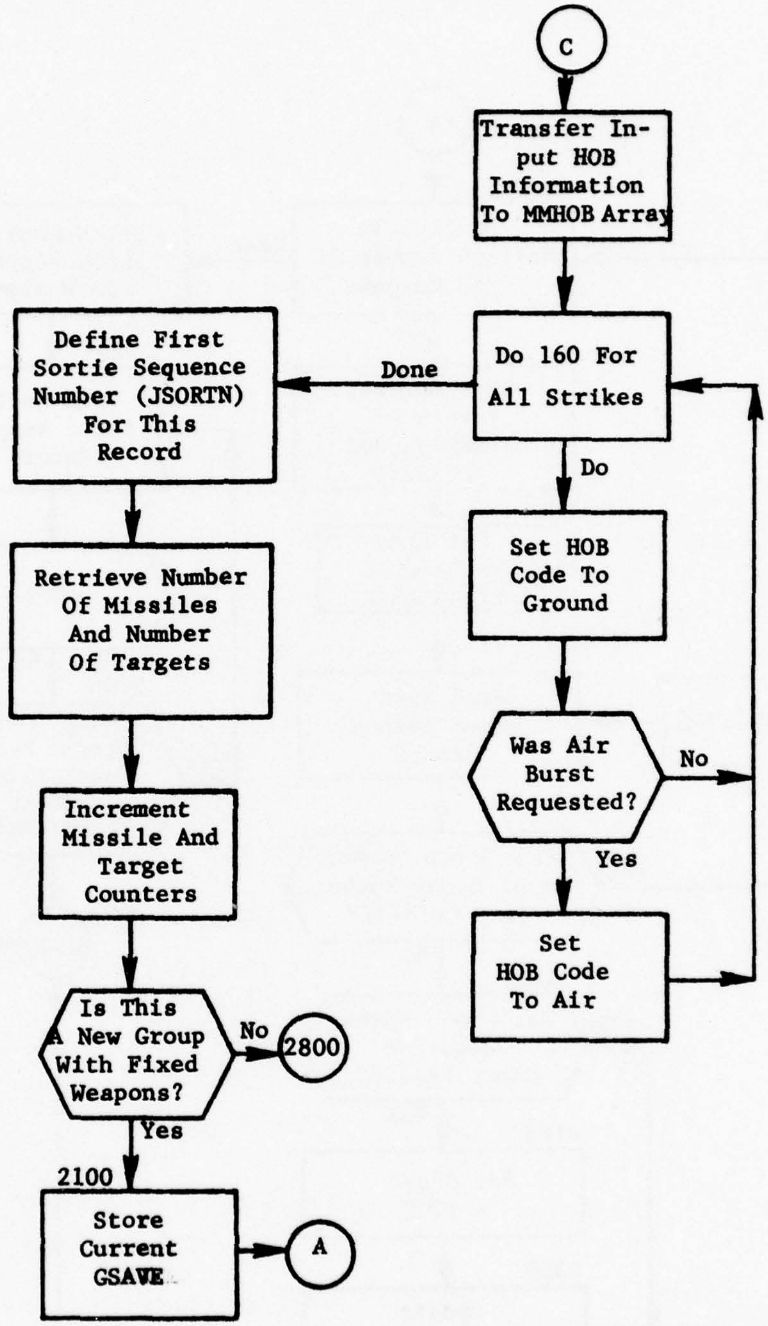


Figure 129. (Part 2 of 5)

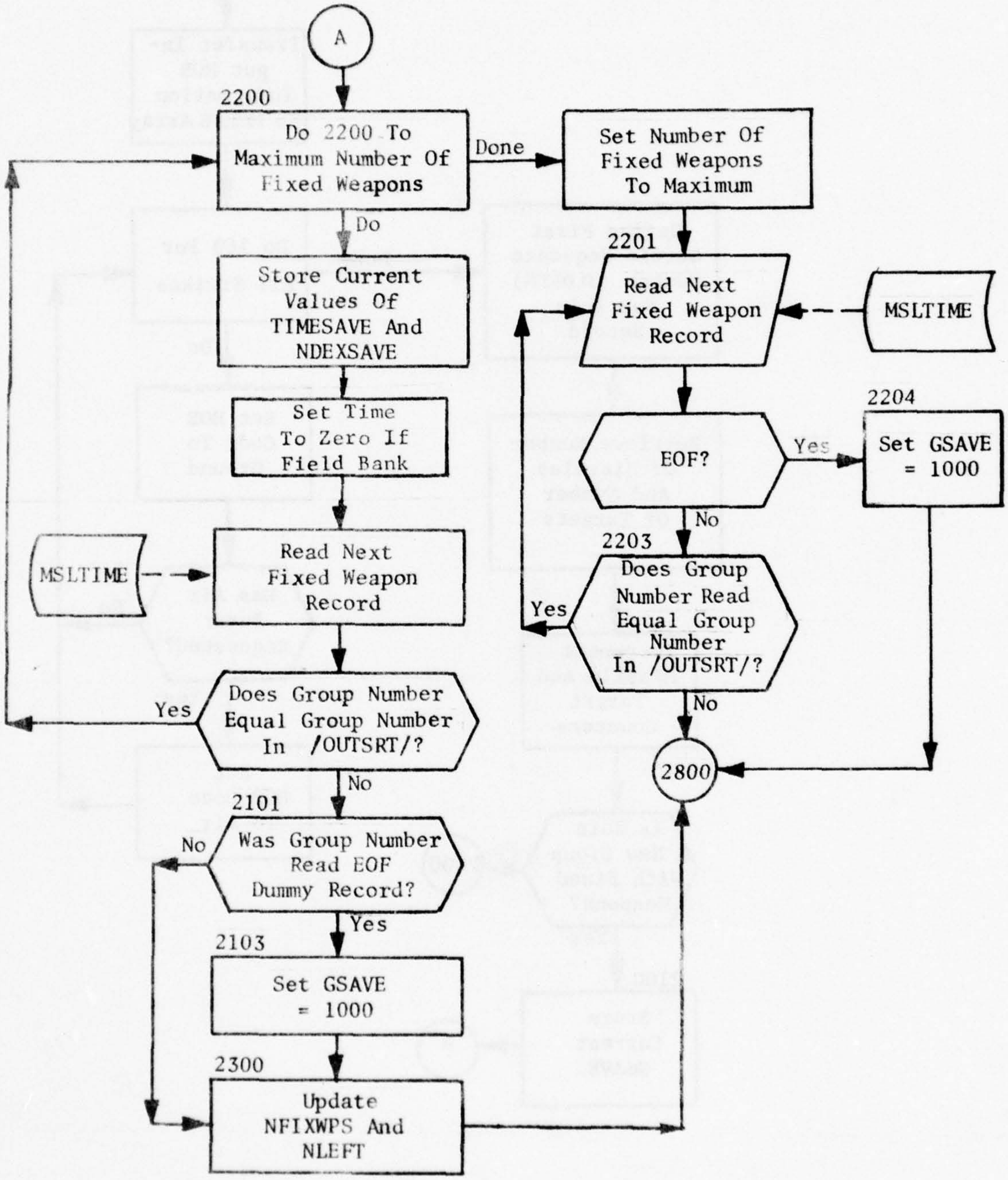


Figure 129. (Part 3 of 5)

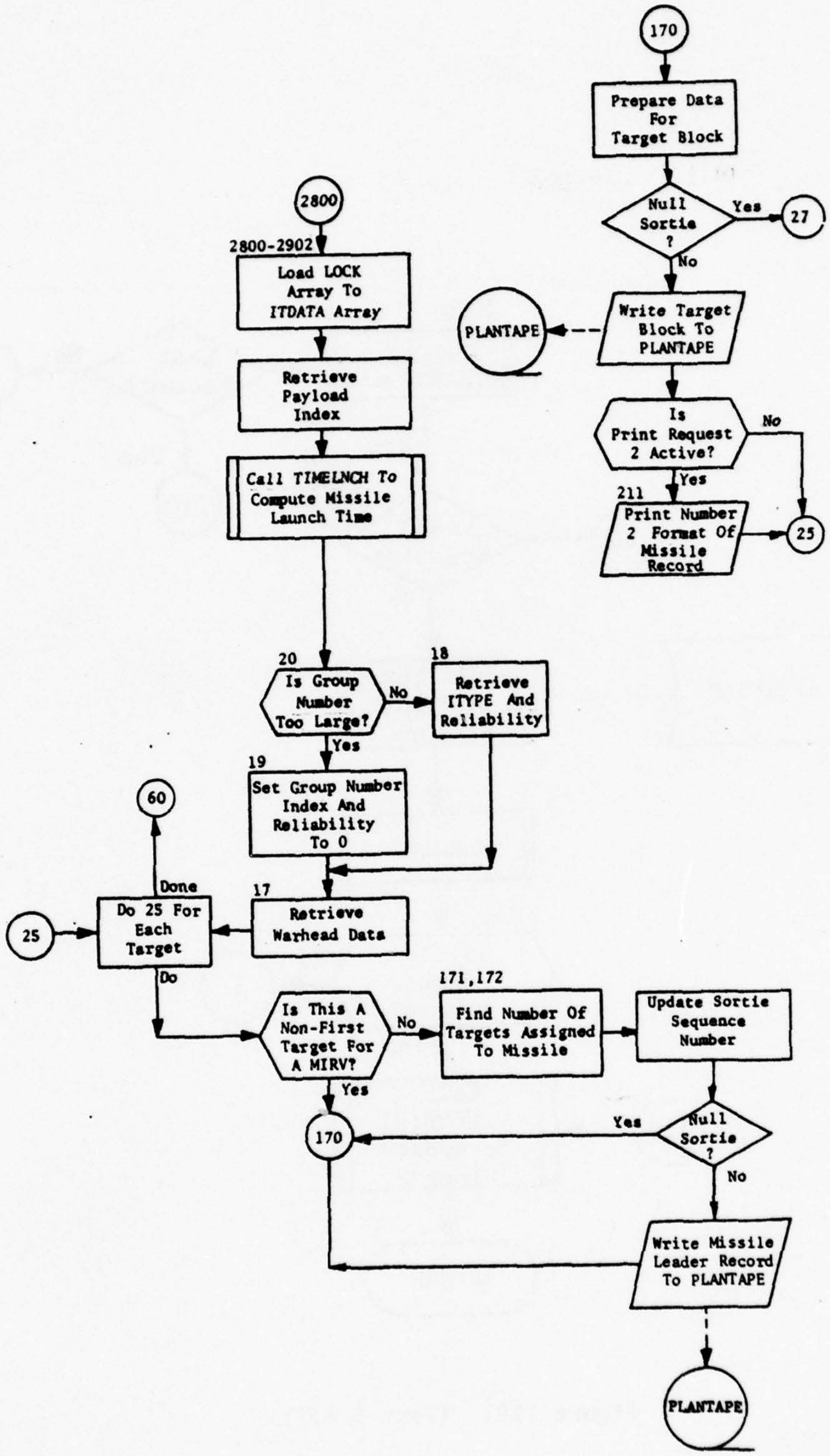


Figure 129. (Part 4 of 5)

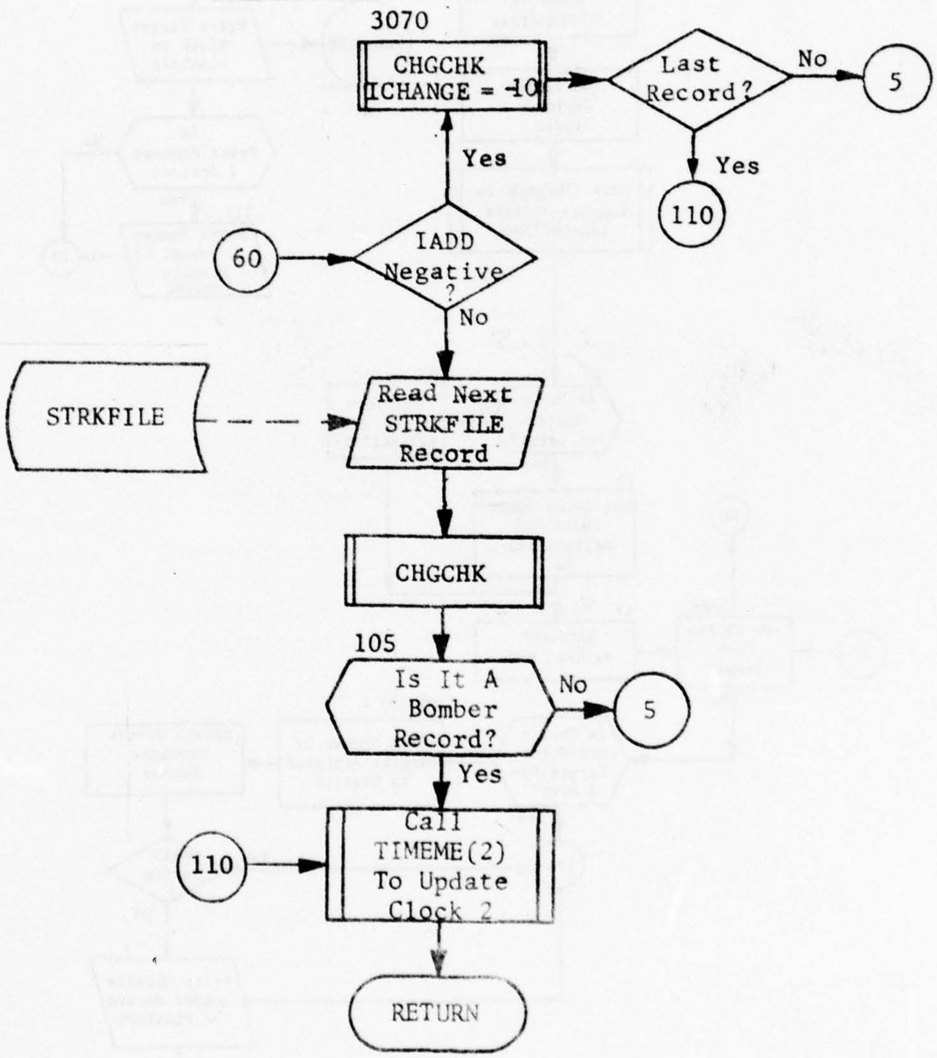


Figure 129. (Part 5 of 5)

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

4.6.2.18 Subroutine POST

PURPOSE: To enter the event type, event location, and place code to the arrays in common /DINDATA/.

ENTRY POINTS:

POST
POST4 (Refuel event)

POST8 (Local Attrition event)
POST15 (formerly, Launch Decoy event; now inactive)
POST17 (Change Altitude event)

FORMAL PARAMETERS:

A = Latitude of event
B = Longitude of event
I = Place code for event

COMMON BLOCKS: DINDATA, EVENTS, FILES

SUBROUTINES CALLED: None

CALLED BY: PLAN

Method:

The event type is determined by the entry point used. Subroutine POST increments the /DINDATA/ line counter (MHT) by 1, then enters the event type code in JTP (MHT), the latitude of the event in HLA (MHT), the longitude in HLO (MHT) and the place index in KPL (MHT).

Subroutine POST is illustrated in figure 130.

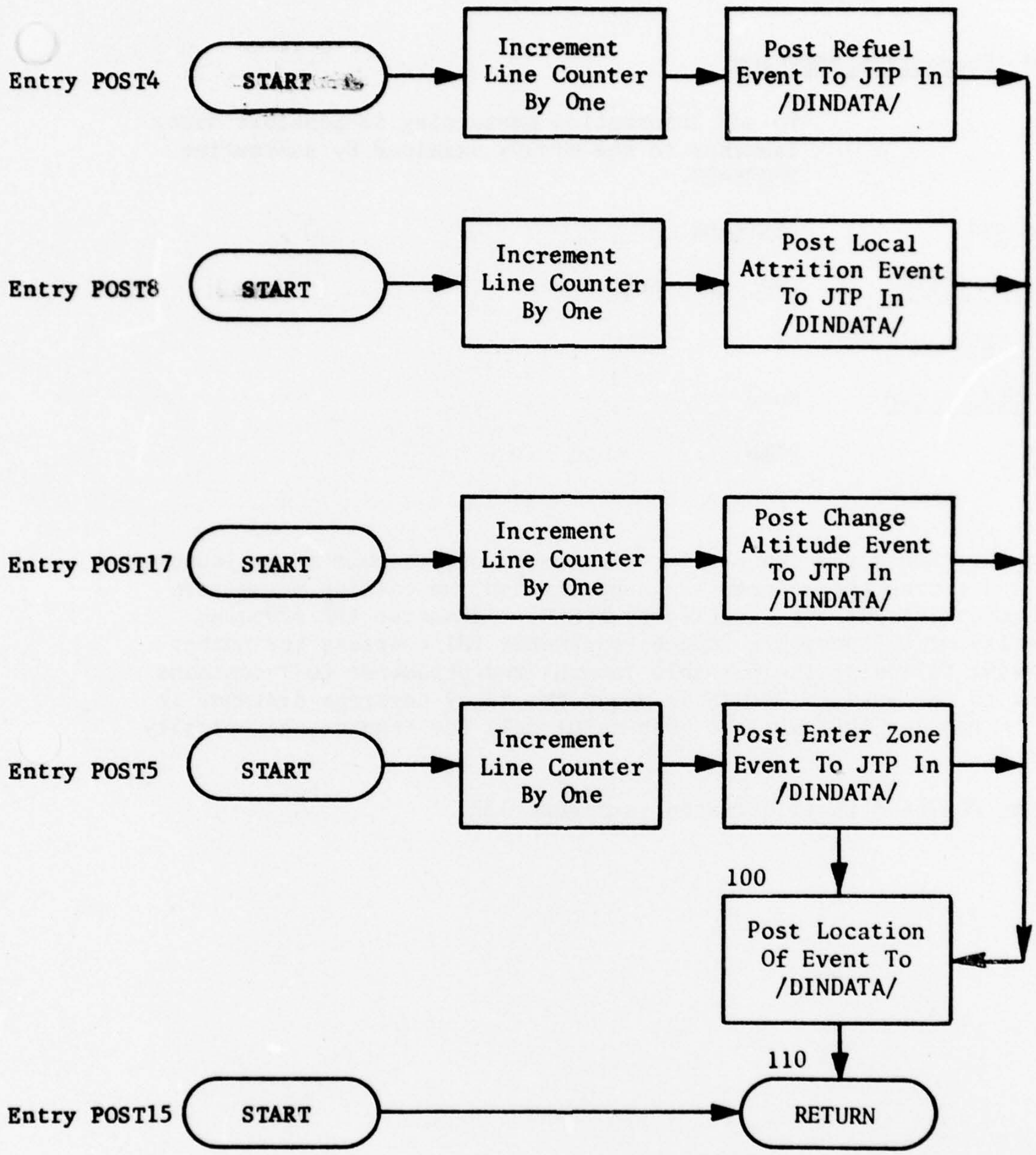


Figure 130. Subroutine POST

4.6.2.19 **Subroutine POSTLAUN**

PURPOSE: To add information pertaining to possible decoy launches to the arrays examined by subroutine DECOYADD.

ENTRY POINTS: POSTLAUN

FORMAL PARAMETERS: LPR, LHT, LDST

COMMON BLOCKS: C1

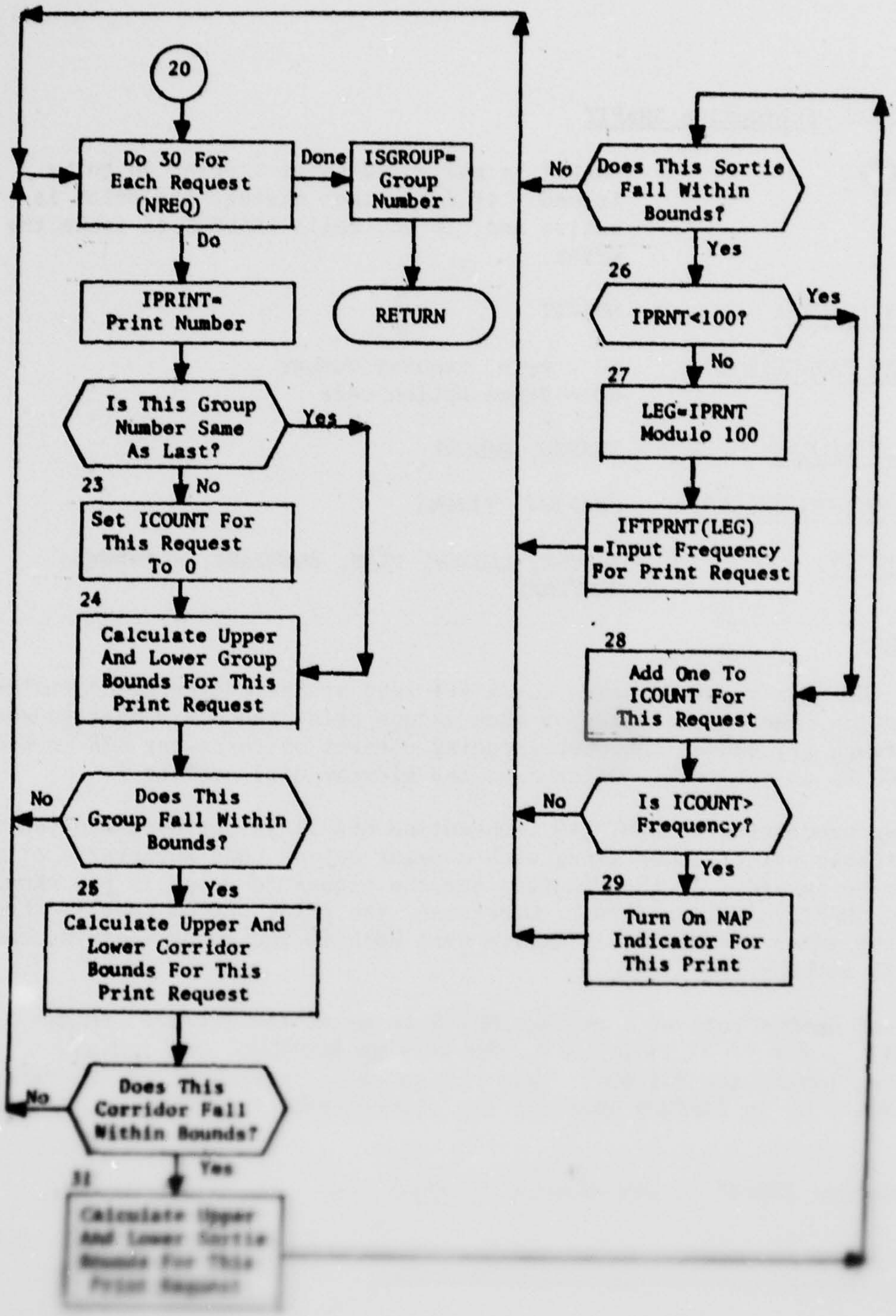
SUBROUTINES CALLED: None

CALLED BY: PLAN

Method:

POSTLAUN increments the counter for the number of possible decoy launches (NPSLN) and stores the information sent through the calling parameters in the appropriate arrays, indexed by NPSLN. Parameter LPR contains the priority of the possible launch; parameter LHT contains the number of the event following the possible launch; and parameter LDST contains the index to the word in DELDIS in which the decoy coverage distance is stored. Parameter LDST will be meaningful only for launches of priority 5, 7, or 8.

Subroutine POSTLAUN is illustrated in figure 131.



20
Do 30 For Each Request (NREQ)
Done
ISGROUP= Group Number
RETURN
Do
IPRINT= Print Number
Is This Group Number Same As Last?
Yes
23
No
Set ICOUNT For This Request To 0
24
Calculate Upper And Lower Group Bounds For This Print Request
No
Does This Group Fall Within Bounds?
25
Yes
Calculate Upper And Lower Corridor Bounds For This Print Request
No
Does This Corridor Fall Within Bounds?
Yes
31
Calculate Upper And Lower Sortie Bounds For This Print Request
No
Does This Sortie Fall Within Bounds?
Yes
26
IPRNT<100?
Yes
No
27
LEG=IPRNT Modulo 100
IFTPRNT(LEG) =Input Frequency For Print Request
28
Add One To ICOUNT For This Request
No
Is ICOUNT> Frequency?
Yes
29
Turn On NAP Indicator For This Print

4.6.2.22 **Subroutine SNAPIT**

PURPOSE: SNAPIT is called whenever a print is to be issued. It determines whether the print is active and, if so, calls SNAPOUT to issue the print.

ENTRY POINTS: SNAPIT

FORMAL PARAMETERS: IO = Print request number
NO = Print option code

COMMON BLOCKS: SNAPON, WAROUT

SUBROUTINES CALLED: SNAPOUT, TIMEME

CALLED BY: ADJUST, LAUNCH, PLAN, PLANTANK, PLANTMIS, PLNTPLAN

Method:

When the user's print option cards are read at the beginning of PLNTPLAN execution, the encountering of each unique print request number (numbered 1 through 15) causes the corresponding element of the array NAP in block SNAPON to be set to 3. Otherwise, the element will contain 1.

During processing of PLNTPLAN, subroutine SNAPIT is called and given the applicable print number along with a print option code whenever a print is to be issued. If the NAP flag for the requested print is not equal to 3, SNAPIT simply returns. Otherwise, the print option code is checked for the value 1, SNAPOUT is called with both IO and NO parameters, and SNAPIT returns.

A print option code of 1 causes SNAPIT to print the message "PRINT NUMBER _, and if it is print 4, the message BOUNDARY SNAP before calling subroutine SNAPOUT. This option code, regardless of its value, is passed on to SNAPOUT where it has significance for prints 4, 11, and 15.

Subroutine SNAPIT is illustrated in figure 134.

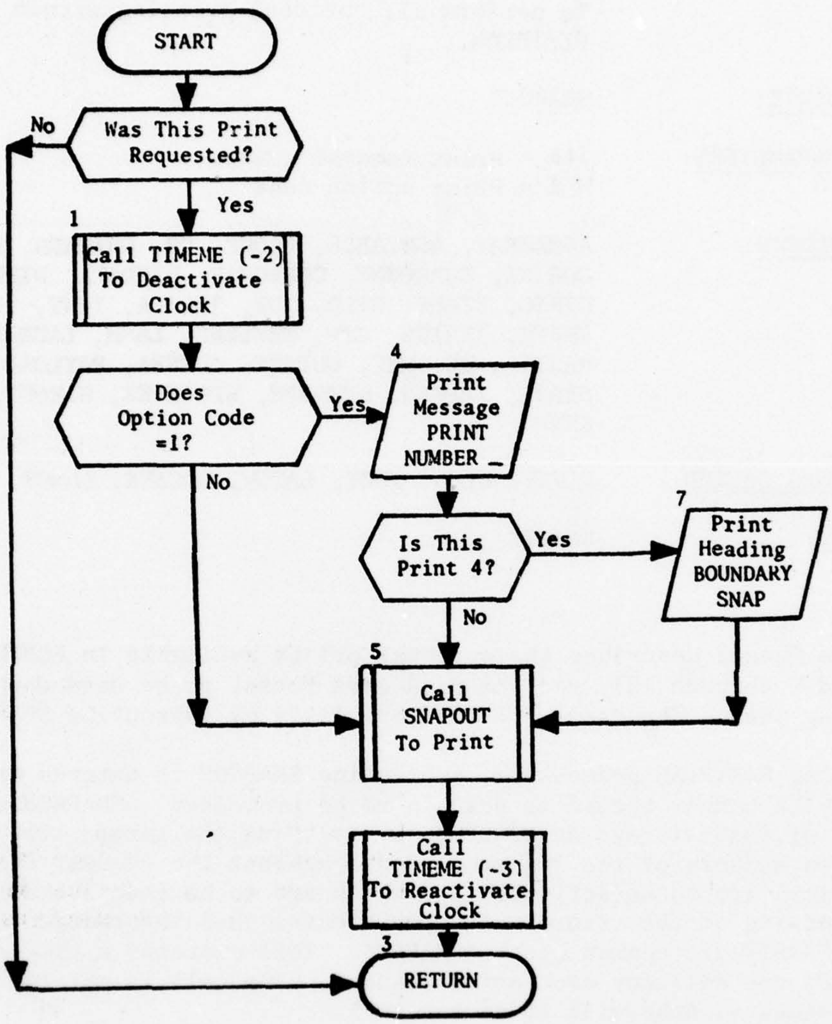


Figure 134. Subroutine SNAPIT

4.6.2.23 Subroutine SNAPOUT

PURPOSE: To perform all optional printing within PLNTPLAN.

ENTRY POINTS: SNAPOUT

FORMAL PARAMETERS: ILK = Print request number
MLK = Print option code

COMMON BLOCKS: ASMARRAY, ASMTABLE, BLOCK, C9, CHANGES, CONTROL, CONTR1, CORCOUNT, CORRCHAR, DINDATA, DINDT2, DISTC, FILES, HILO, IDP, INDATA, IOUT, IRF, IRFTK, ISRTNS, ITP, KEYLENG, LASM, LAUNSNAP, MASTER, MH, MH2, OUTSRT, OUTSRA, PAYLOAD, PLANTAPE, SPASM, TANKER, WPNGRPX, WPNTYPEX, WAROUT, PAYDATA, KEYM

SUBROUTINES CALLED: DISTF, GLOG, IGET, LATCV, LOCATE, LONCV, TIMEME

CALLED BY: SNAPIT

Method:

The Users Manual describes the optional prints available in PLNTPLAN (numbered 1 through 15), and the data card format to be used when requesting them. The cards are read initially by subroutine SNAPCON.

Then during PLNTPLAN processing, subroutine SNAPCON is entered as each new STRKFILE bomber record is read in to be processed. SNAPCON scans the list of requests and determines, by matching the group, corridor, and sortie numbers of the incoming record against the request list, which prints are to be activated and which are to be inactive during the processing of the record. It communicates this information to subroutine SNAPIT via common block /SNAPON/. This contains a 15-word array NAP, one cell for each print request. The cell is set to 3 for active requests; otherwise it is set to 1.

The subroutine SNAPIT is called during the PLNTPLAN program or its subroutines wherever a particular print might possibly be issued. For example, SNAPIT is called upon to print the detailed plan immediately after this plan has been completed. SNAPIT then checks cell 3 of NAP and issues the print only if the cell is set to 3. It calls on subroutine SNAPOUT to do the actual printing. This separation of

THE CONTENTS OF THESE PAGES INTENTIONALLY DELETED

THE CONTENTS OF THIS PAGE INTENTIONALLY DELETED

THE CONTENTS OF THESE PAGES INTENTIONALLY DELETED

THIS PAGE INTENTIONALLY LEFT BLANK

Table 43. Sortie Event Plotting Symbols

EVENT TYPE	EVENT	ACRONYM	SYMBOL
2	Launch bomber	LAUNB	☆
4	Refuel	LEREFUEL	X
8	Drop bomb	LOCLATTR	□ (number by sequential code)
14	Launch ASM	LAUNASM	X (and broken line to target)
15	Launch decoy	LAUNDCOY	D
16	Recovery	LANDHO	
13	Abort	LABORT	*
18	Go to high altitude	IGOHI	HI
19	Go to low altitude	IGOLOW	LO
20	Dogleg	LEGDOG	
10	Change time	ITIME	

6.4 Concept of Operation

Program PLOTIT is designed to produce plots of all or any specified sorties contained in the PLANTAPE files. The user may choose his plotting options via punched card input. When PLOTIT starts executing, it reads in the plot label, plot size and number of cases to plot. For each case which is to be plotted the necessary control parameters are read in, such as Blue or Red side, plot origin, number of graphs per plot, number of sorties to be plotted, etc. For each graph the projection parameters and desired ticmarks are read in. As these user-specified parameters are input, they are printed on-line. Then the map type indicator and map projection parameters are input. Subroutine PIECEIT is called to initialize the plot pen, and subroutine PROJCT3 to compute the projection variables necessary for the conversion of latitudes and longitudes to inches and to convert the map corner coordinates. Then the ticmarks are set up in the TICX and TICY arrays and checked by subroutine CHKTIK to determine if they fall within the margins of the graph.

If plotting of specific sorties rather than all sorties in the file is indicated, the parameters identifying the desired sorties are read.

Sorties do not have to be plotted in the order they appear on the PLANTAPE, as an indicator may be set to rewind the tape and search for the desired sortie. The plot and graph numbers are printed and the PLANTAPE is positioned to input the data for the specified sortie. Subroutine SUBREAD is called to save the sortie parameters as they are input for later use by the plotting routine. When all sortie records for a plot have been read, subroutine SUBPLOT is called to perform the actual plotting of the data. This is repeated for all sorties which are to be plotted.

A description of the common blocks used by program PLOTIT is given in table 44. The flow of the program is illustrated in figure 160.

Table 44. Program PLOTIT Common Blocks (Part 1 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
EVENTS	LAUNM	Launch missile code
	LAUNB	Bomber launch code
	LEREFUEL	Refuel code
	LOCLATTR	Local attrition or drop bomb event code
	LAUNASM	Launch ASM event code
	LAUNDCOY	Launch decoy event code
	LANDHO	Recovery event code
	LOHI	Change altitude event code
	MISSATTR	Missile attrition event code
	LEGDOG	Dogleg event code
	LABORT	Abort event code
	LENTEREF	Enter refuel area event code
	LEAVEREF	Leave refuel area event code
	IGOHI	Go to high-altitude event code
	IGOLOW	Go to low-altitude event code
	ITIME	Change time event code
	KSMTGT	ASM flight-time indicator
INMAP	INMAP	Indicator if MAPEDGE has been executed
	XLBL	Plot point number
LMBRT	FL	Lambert projection parameter
	FK	Lambert projection parameter
PLNTAPE	KPL	Array of indices for bases, zones, targets or number of decoys launched

Table 44. (Part 2 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
PLNTAPE (cont.)	JTP	Array of sortie event types
	HLA	Array of sortie event latitudes
	HLO	Array of sortie event longitudes
	IWH	Array of warhead type indices
	CMT	Array of cumulative time to event
	MHLOW	Lower plot markers for sortie
	MHHI	Upper plot markers for sortie
	MYGROUP	Group number
	MYCORR	Penetration corridor number
	IOUTSRT	Bomber sortie number
	LTOT	Total number of bomber events
	LPLAN	Number of planned bomber events
PLTARRYS	ILOW	Current index into arrays where information is saved
	SUBHLA	Array to save event latitudes
	SUBHLO	Array to save event longitudes
	SUBJTP	Array to save event types
	SUBKPL	Array to save base, zone, target indices or number of decoys launched
	IWARHD	Array to save warhead type indices
	CTIME	Array to save cumulative time of event
	MYGRUP	Array of group numbers
	MYCORR1	Array of penetration corridor numbers
	IOUTSORT	Array of bomber sortie numbers